Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Donaire Alós, Gerard

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Aplicació per la Gestió de Cues de Telèfons**

Director/a: **Granollers Saltiveri, Antoni**

Presentació

Mes: Juliol

Any: 2019

Índex

1 Resum **1**

2 Abstract **2**

3 Agraïments **3**

4 Paraules Clau **3**

5 Key Words **3**

6 Llista d'Acrònims **3**

7 Introducció **4**
7.1 Necessitat d'aquest projecte 4
7.2 Justificació i Interès del tema 5
7.3 Vinculació del tema escollit amb les competències del Grau 5
7.4 Estructuració del projecte 6

8 Metodologia **7**
8.1 Scrum 7

9 Objectius del treball final de grau **8**
9.1 Objectius Generals 8
9.2 Objectius Específics 8

10 Estat de l'Art **9**
10.1 Història d'Ilerna Online 9
10.2 Ilerna Online en l'Actualitat. 9

11 Desenvolupament del Projecte **11**
11.1 Iteració 1 11
11.1.1 Definició dels requeriments 11
11.1.1.1 Requeriments Funcionals 11
11.1.1.2 Requeriments No Funcionals 14
11.1.2 Planificació 15
11.1.3 Disseny 16
11.1.4 Implementació 19
11.1.4.1 Routes 19
11.1.4.2 Api.php 19
11.1.4.2.1 Rutes Students 19
11.1.4.2.2 Rutes Agents 23
11.1.4.2.3 Rutes Queues 25
11.1.4.2.4 Ruta AgentStudent 25
11.1.4.2.5 Rutes Cursos 25
11.1.4.2.6 Rutes Blacklist 26
11.1.4.2.7 Rutes Semester 27
11.1.4.2.8 Rutes Token 27
11.1.4.3 Models 28
11.1.4.3.1 AgentStudent.php. 28
11.1.4.3.2 Agent.php 28
11.1.4.3.3 AgentTimeOff.php 28
11.1.4.3.4 Blacklist.php 28
11.1.4.3.5 Course.php 28

11.1.4.3.6 Enroll.php . . . 29
11.1.4.3.7 Queue.php . . . 29
11.1.4.3.8 Semester.php . . . 29
11.1.4.3.9 Student.php . . . 29
11.1.4.3.10 StudentPhone.php . . . 29
11.1.4.3.11 Template.php . . . 29
11.1.4.3.12 User.php . . . 29
11.1.4.4 Controllers . . . 30
11.1.4.4.1 AgentController.php . . . 30
11.1.4.4.2 AgentStudentController.php . . . 32
11.1.4.4.3 BlacklistController.php . . . 32
11.1.4.4.4 Controller.php . . . 32
11.1.4.4.5 CourseController.php . . . 33
11.1.4.4.6 QueueController.php . . . 33
11.1.4.4.7 StudentController.php . . . 33
11.1.5 Tests de requeriments . . . 35
11.2 Iteració 2 . . . 40
11.2.1 Definició dels Requeriments . . . 40
11.2.1.1 Requeriments Funcionals . . . 40
11.2.2 Planificació . . . 41
11.2.3 Disseny . . . 42
11.2.4 Implementació . . . 42
11.2.4.1 Rutes Shops . . . 43
11.2.4.2 Models . . . 43
11.2.4.2.1 Shop.php . . . 43
11.2.4.3 Controllers . . . 43
11.2.4.3.1 ShopController.php . . . 43
11.2.4.4 Classes per a les migracions de BD. . . . 44
11.2.5 Tests de requeriments . . . 46

12 Conclusions **48**

13 Treball Futur **49**

14 Referències bibliogràfiques **50**
14.1 Consultes sobre PHP . . . 50
14.2 Consultes sobre SQuirrel SQL Client, MySQL i SQL . . . 50
14.3 Consultes sobre PhpSpreadSheet . . . 51
14.4 Consultes sobre Laravel Eloquent . . . 52
14.5 Consultes sobre Servidor Apache . . . 53
14.6 Consultes sobre HTML . . . 53
14.7 Consultes sobre PhpStorm . . . 53
14.8 Consultes sobre Metodologia Agile . . . 53
14.9 Consultes sobre Programació per Capes . . . 54

15 Annex **55**
15.1 Definicions . . . 55
15.2 Avantatges i Inconvenients de la Metodologia *Agile* . . . 58
15.2.1 Avantatges . . . 58
15.2.2 Inconvenients . . . 58
15.3 Funcionalitats BackEnd i FrontEnd . . . 59
15.4 Programació per capes. . . . 59
15.4.1 Capes i nivells. . . . 60

Índex de figures

1 Primera versió del Diagrama UML sobre la base de dades de l'aplicació. 17
2 Segona versió del Diagrama UML sobre la base de dades de l'aplicació. 18
3 Menú de l'aplicació. 35
4 Informació sobre els *agents*. 36
5 Informació sobre els *agents* i els alumnes que no poden estar junts. 36
6 Pantalla d'informació sobre els cursos i botons de guardar i tirar endarrere. 36
7 Pantalla d'informació sobre les cues. 37
8 Pantalla d'assignacions amb els filtres demanats en l'iteració 1. 37
9 Pantalla de creació *agents*. 37
10 Pantalla de modificació d'*agents*. 38
11 Pantalla d'eliminació d'*agents*. 38
12 Excel exportat amb els camps corresponents. 38
13 Pantalla de creació de *Blacklist*. 39
14 Opció d'eliminació d'una *Blacklist*. 39
15 Pantalla de creació de Cues. 39
16 Tercera versió del Diagrama UML sobre la base de dades de l'aplicació. 42
17 Pantalla de modifiació de *Shops*. 46
18 Menú amb l'opció Shops i funcionalitat implementada. 46
19 Pantalla d'assignacions corresponent a l'iteració 2. 46
20 Pantalla corresponent a l'opció *Shops*. 47
21 Esquema de la ubicació del programari intermediari (*middleware*). 56
22 Diagrama UML sobre els Models i els Controladors de l'aplicació. 61
23 Diagrama UML sobre la base de dades de l'aplicació. 62

1 Resum

L'empresa Ilerna Online representa un centre d'estudis de formació professional a distància i oficial. Ilerna centra tots els seus esforços en promoure una Formació Professional professionalitzadora i integradora, reforçant els valors tradicionals que sempre els ha identificat com són la cultura de l'esforç, l'atenció personalitzada, la internacionalització, les noves tecnologies i, sobre tot, la innovació constant en tots els àmbits, per tal d'aconseguir un gran repte i convertir-lo en una realitat immediata.

Aquesta empresa consta de diversos departaments dels quals sols esmentaré dos ja que són els que estan més implicats en el projecte que explicaré més endavant. Aquests departaments corresponen al departament d' IT (representa el departament on treballen els professionals encarregats en la informàtica) i el departament de *Contact* (representa el departament on treballen els professionals que s'encarreguen de l'atenció al client). Aquests professionals que treballen al departament de *Contact* els anomenarem *agents*.

Inicialment, el departament de Contact no tenia un gestionament eficaç i ordenat sobre les trucades que realitzaven els alumnes. De manera que quan un alumne trucava, aquesta mateixa trucada no anava dirigida a un agent en particular sinó al primer agent que l'agafava. Llavors, la idea principal que demanava aquest departament era la possibilitat de realitzar una aplicació web per tal de poder assignar agents en unes cues [1] determinades i de distribuir els alumnes de manera equilibrada entre aquests agents tenint en compte una sèrie de requeriments.

Per tal de realitzar aquest projecte i durant tot el recorregut d'aquest document, he hagut de debatre amb els meus companys de feina i informar-me tant de les noves eines i tecnologies a utilitzar, com dels requeriments que volia el client.

Un altre punt important, a fi de dur a terme aquest projecte de software, és que hem necessitat una organització racional del treball i hem tingut d'armar-nos amb criteris duradors per a prendre decisions coherents al llarg del cicle de vida del projecte. Això és, el que en el món de l'arquitectura del software anomenem metodologia. En el nostre cas hem fet ús d'una metodologia anomenada *Agile*. Aquesta metodologia és aquella que permet flexibilitzar o acomodar, d'alguna manera, el projecte en el que s'aplica. Es refereix a un grup de metodologies aplicades en la creació de *software* que basa el seu desenvolupament en un cicle iteratiu. Es va decidir usar aquest tipus de metodologia amb la intenció de disposar una millor organització del treball i una constància temporal.

Per tal d'implementar *Agile* hem seguit la metodologia més popular anomenada *Scrum*. Aquesta metodologia està aplicada en un marc de treball on interessa anar traient feina al client en uns intervals de temps predefinits, iteracions, que permeten flexibilitzar la feina que s'està realitzant i cal remarcar l'ús d'esdeveniments de *Scrum* per a regular les reunions de l'equip. Com ara, *Daily Scrum, Sprint Review, Sprint Planning* o *Sprint Restrospective* els quals estan explicats a la secció **8**.

[1] representen una alineació de trucades

2 Abstract

The company Ilerna Online represents a vocational training center at distance and official. Ilerna focus all his efforts at promoting a Professional Formation, reinforcing the traditional values that always has identified the ones how are the culture of the effort, the attention customised, the internationalization, the new technologies and, above all, the constant innovation in all areas, for such to achieve a big challenge and convert it at an immediate reality.

This company consists of several departments of which I will mention only two because they are the ones that are most involved in the project that I will mention later. These departments correspond to the IT department (it represents the department where the professionals in charge of computer science work) and the Contact department (it represents the department where the professionals who are in charge of customer service work). These professionals who work in the department of Contact we will call them agents.

Initially, the Contact Department did not have an efficient and orderly management of the calls made by the students. So when a student called, this same call was not addressed to a particular agent but the first agent who took it. Then, the main idea asked by this department was the possibility of making a web application in order to be able to assign agents in a certain queues [2] and distribute the students in a balanced way among these agents taking into account a series of requirements.

In order to carry out this project and throughout the course of this document, I had to discuss with my colleagues and inform me about the new tools and technologies to use, as well as the requirements that the client wanted to be able to carry out this project.

Another important point in order to carry out this software project is that we needed a rational work organization and we had to arm ourselves with lasting criteria to make coherent decisions throughout the life cycle of the project. That is, what in the world of software architecture we call methodology. In our case we have used a methodology called Agile.This methodology is one that allows to flexibilize or accommodate, in some way, the project in which it is applied. It refers to a group of methodologies applied in the creation of software that bases its development in an iterative cycle. It was decided to use this type of methodology with the intention of having a better organization of the work and a temporal constancy.

In order to implement Agile, we have followed the most popular methodology called Scrum. This methodology is applied in a framework of work where it is interesting get the work out to the customer in a predefined time intervals, iterations, which allow for flexibility the work that we are doing and it is necessary to emphasize the use of events of Scrum to regulate the meetings of the team. Like Daily Scrum, Sprint Review, Sprint planning or Sprint restrospective that are explained in the section **8**.

[2] represents a call alignment

3 Agraïments

M'agradaria expressar el meu més profund agraïment tant als professionals del departament d'IT de l'empresa Ilerna Online com al Toni Granollers (com a tutor d'aquest treball) per donar-me la seva paciència, temps i dedicació la qual cosa ha fet possible la realització d'aquest treball.

Gràcies per la seva orientació i ajuda en aquest intens camí.

4 Paraules Clau

Cues, curs, model, vista, agent, telèfons, semestre, estudiant, matricular, assignació, controlador, distribució, llista negra, aplicació web, atenció al client, agent de baixa, agent especial.

5 Key Words

Queues, course, model, view, agent , telephones, semester, student, enroll, allocation, controller, distribution, black list, web application, costumer service, agent time off, special agent.

6 Llista d'Acrònims

IT: Information Technology. (Tecnologia de la informació).
PHP: Hypertext Pre-Processor. (Llenguatge de Programació Interpretat).
IDE: Integrated Development environment. (Entorn de desenvolupament integrat).
HTML: HyperText Markup Language. (Llenguatge de Marcat d'Hipertext).
MySQL: My Structured Query Language. (Llenguatge de Consulta Estructurat).
SQL: Structured Query Language. (Llenguatge d'interrogació estructurat).
MVC: Model-View-Controller. (Model-Vista-Controlador)·
PHP IDE: Hypertext Pre-Processor Integrated Developed Environment.
(Entorn Desenvolupat Integrat del Llenguatge de Programació Interpretat).
FCT: Formació Centres de Treball.
JSON: JavaScript Object Notation.
XML: Extensible Markup Language.
HTTP: Hypertext Transfer Protocol. (Protocol de transferència d'hipertext.)
API: Application Programming Interface. (Interfície de programació d'aplicacions.)

7 Introducció

Aquest projecte es basa en el desenvolupament d'una aplicació web per a la gestió de cues [3] de telèfons per a al departament de ***Contact*** de l'empresa Ilerna Online. Aquesta empresa representa un centre d'estudis de formació professional a distància i oficial. Si es vol saber més detall sobre aquesta empresa anar a la secció **10**.

7.1 Necessitat d'aquest projecte

Aquesta aplicació sorgeix de la necessitat que té el departament de *Contact* de l'empresa *Ilerna Online* per tal de poder assignar agents [4] en unes determinades cues de telèfons.

En un inici aquest departament no tenia un gestionament eficaç i ordenat sobre les trucades que realitzaven els alumnes. De manera que quan un alumne trucava, aquesta mateixa trucada no anava dirigida a un agent en particular sinó al primer agent que l'agafava. Llavors, la idea principal que es demanava era la possibilitat de realitzar una aplicació web per tal de poder assignar agents en unes cues determinades i de distribuir els alumnes de manera equilibrada entre aquests agents tenint en compte una sèrie de requeriments (els quals els podem veure tots a la subsecció **11.1.1** i **11.2.1**). En aquest cas esmentaré el requeriment més important corresponent a la distribució dels alumnes entre els agents (Requeriment **R.8.1**). Depenent d'aquest requeriment uns alumnes es distribuiran a un agent especial o bé a un agent.

Un agent especial representa un agent el qual s'encarrega dels alumnes que són *premium* [5] i/o que pertanyen a centres col·laboradors [6]. Per altra banda, els agents són els encarregats de tenir tots els altres alumnes que no són *premium* i tampoc pertanyen a centres col·laboradors. De manera que a un agent especial (en el nostre cas del projecte sols tindrem 1 agent especial) se li distribuiran tots aquells alumnes que siguin *premium* i/o pertanyin a centres col·laboradors. Pel que fa a la distribució d'alumnes als agents se'ls hi assignaran la resta d'alumnes que no són *premium* i que tampoc pertanyen a centres col·laboradors, però tenint en compte un ordre de criteris de més a menys prioritat el qual és el següent:

Primerament el criteri amb més prioritat, mirem si l'alumne està cursant a Formació Centres de Treball, FCT, (representen alumnes d'Ilerna Online que fan pràctiques a una empresa) i observem quins agents tenen menys càrrega d'alumnes que pertanyen a FCT. És a dir, mirem quin agent té menys alumnes de FCT. Si no hi ha empat i trobem un agent el qual és el que té menys alumnes de FCT se li assigna l'alumne. En cas d'empat, es miraria el segon criteri el qual representa: d'aquests agents (els que han empatat), es mira el nombre total d'alumnes que té cadascun. Si no hi ha empat i trobem un agent el qual és el que té el número total d'alumnes menor, se li assigna l'alumne. En cas d'empat es miraria el tercer criteri el qual representa el grup de botiga [7] a la qual pertany aquell estudiant i es mira pels agents que han empatat quin agent té menys alumnes d'aquell grup de botiga. Si no hi ha empat s'assigna l'alumne a l'agent que té menys alumnes en el grup de botiga que correspon l'alumne en qüestió. En cas d'empat s'arribaria al quart criteri el qual correspon a la família del curs al qual pertany l'alumne i es mira pels agents que han empatat quin agent té menys alumnes. Si no hi ha empat s'assigna l'alumne a l'agent que té menys alumnes referent a la família del curs de l'alumne en qüestió. Finalment si encara hi hagués empat entre agents, realitzaríem un *random* (selecció aleatòria) i s'assignaria l'alumne a un agent aleatori entre els agents que havien empatat en l'últim criteri.

Aquest projecte constarà d'objectius generals i específics els quals els podem veure a la subsecció **9.1** i **9.2** corresponents.

[3] representen una alineació de trucades. Una trucada pot correspondre a un número de telèfon diferent. Per tant, per saber si la trucada correspon a un alumne de Ilerna Online o algú extern l'agència encarregada de l'enrutament de les trucades, anomenada MasVoz, representa cada cua amb un identificador (id) en concret.

[4] els quals representen professionals de la pròpia empresa que s'encarreguen de l'atenció dels clients.

[5] alumnes que paguen addicionalment per un privilegi

[6] és a dir, si un alumne pertany a un altre centre o botiga diferent de *Ilerna Online*.

[7] representa el nom de l'agrupació de les botigues independentment de la seva ubicació.

Existeixen diferents models de procés per l'Enginyeria de *Software*. Cadascun d'aquests models pretén , d'una manera o altra, proporcionar un ordre el més simple possible al complicat procés de desenvolupament de *software*. Per al cas d'aquest projecte, és necessari apegar-se el més possible a un d'aquests models amb la finalitat de tenir una organització d'activitats que es plantegen en base d'etapes lògiques i interconnectades entre si. El model d'enginyeria de *software* que utilitzarem segueix el model *Agile*, el qual està descrit en la secció corresponent a **Metodologia**.

7.2 Justificació i Interès del tema

He escollit aquest tema ja que trobo interessant el fet d'aprendre a dissenyar i estructurar una aplicació web en una empresa. A més, quan em van explicar aquest projecte per primer cop m'agradava la idea de realitzar aquesta aplicació perquè trobava interessant el fet de com distribuir per a cada agent els estudiants de manera equilibrada i veure com es treballava en equip en una empresa. També tenia ganes de realitzar algun projecte web com aquest. A més, la idea de saber com gestionar una aplicació com aquesta crec que em servirà per guanyar experiència i poder utilitzar-ho més endavant en un futur proper.

7.3 Vinculació del tema escollit amb les competències del Grau

Al llarg de tot el recorregut emprat a la Universitat de Lleida realitzant el grau d'Enginyeria Informàtica, vaig escollir fer la branca més similar en Intel·ligència Artificial tenint en compte que trobava interessant la idea de saber optimitzar i realitzar algorismes. En conseqüència d'aquest fet, en aquests anys a la Universitat no vàrem tocar massa l'apartat de web. Per tant m'agradaria, abans d'acabar el grau, saber el màxim possible sobre aquest tema. Per aquest motiu volia aprendre també sobre la branca de *software*, volia aprendre i realitzar un projecte web prou voluminós per tal d'assolir el màxim de coneixements sobre web per així poder-me defensar en un futur proper.

7.4 Estructuració del projecte

Aquesta documentació es basarà en les següents seccions:

- **Metodologia:** en aquest apartat descriuré la metodologia d'estudi, que és part del pla d'investigació. Explicaré quina metodologia hem fet servir per dur a terme el projecte.
- **Objectius del treball final de grau:** En aquest apartat detallaré els objectius tant generals com específics sobre el projecte.
- **Estat de l'Art:** en aquest apartat parlarem sobre l'empresa Ilerna Online per tal d'informar i familiaritzar al lector sobre aquesta empresa.
- **Desenvolupament del Projecte:** En aquest apartat parlaré sobre les dues iteracions que hem realitzat per dur a terme el projecte i definiré els requeriments corresponents per a cada iteració. Tant la primera iteració com la segona es dividirà en:
  - **Definir els requeriments:** representen el conjunt de peticions que demana l'usuari. Aquestes peticions ajuden al desenvolupador del *software* per comprendre completament la naturalesa dels programes que s'han de construir per desenvolupar l'aplicació.
  - **Planificació:** és on esmentarem la planificació del projecte emprat per tal de començar a donar una estructura al projecte. És la part més important ja que les següents fases partiran sobre aquesta.
  - **Disseny:** es refereix a l'establiment de les estructures de dades, l'arquitectura general del *software*, representacions d'interfícies i algorismes.
  - **Implementació:** fa referència en convertir una idea, un esquema, etc., en un objecte o en un procés informàtic en estat operatiu.
  - **Tests:** seran proves amb el *feedback*[8] del client per tal de saber si els requeriments s'adapten als demanats per part del client i per veure si hi ha algun error.

  Pel que fa a la primera iteració, és on implementarem bona part dels requeriments. En la segona iteració és on es milloraran i es continuaran les implementacions mitjançant el *feedback* i els problemes obtinguts de l'anterior iteració. També implementarem i comprovarem que s'han dut a terme tant els nous requeriments com els vells esmentats pel client.
- **Conclusions i Opinió Personal:** en aquest apartat explicaré les conclusions a les quals hem arribat abans i després de dur a terme el projecte amb una petita opinió personal del que m'ha semblat el projecte.
- **Treball Futur:** en aquest apartat explicaré fins on hem arribat i el que hauríem d'ampliar del projecte per a que sigui un projecte més complet com ara decisions de disseny de futur.
- **Referències bibliogràfiques:** en aquest apartat es mostraran tots els llocs visitats que m'han ajudat per aclarir idees a l'hora de desenvolupar el projecte.
- **Annex:** en aquest apartat mostraré informació externa com ara definicions per a que el lector pugui saber en tot moment que significa el que s'està dient en aquesta documentació.

[8] representa la reacció, resposta o opinió que ens dóna un interlocutor com a retorn sobre un assumpte determinat.

8 Metodologia

En qualsevol projecte de *software* que vagi més enllà d'una prova de concepte o demostració, necessitarem una organització racional del treball. Tindrem que esforçar-nos en buscar patrons repetibles per agilitzar futurs processos i ens hem d'armar-nos amb criteris duradors per a prendre decisions coherents al llarg del cicle de vida del projecte. Això és, el que en el món de l'arquitectura del *software* anomenem metodologia. Per realitzar aquest projecte hem utilitzat un tipus de metodologia anomenada *Agile*.

Una metodologia ***Agile*** és aquella que permet flexibilitzar o acomodar, d'alguna manera, el projecte en el que s'aplica. Es refereix a un grup de metodologies aplicades en la creació de *software* que basa el seu desenvolupament en un cicle iteratiu. Es va decidir utilitzar aquest tipus de metodologia per a tenir una millor organització del treball i una constància temporal.

8.1 Scrum

Aquesta metodologia està aplicada en un marc de treball on interessa anar traient feina al client en uns intervals de temps predefinits, iteracions, que permeten flexibilitzar la feina que s'està realitzant. En aquesta, hi ha tres tipus de rols:

- ***Product Owner:*** persona encarregada del treball realitzat per l'equip de treball. És l'encarregat d'ocupar-se del *backlog* [9].
- ***Scrum Master:*** és la persona que s'ocupa d'aconseguir guiar a tot l'equip en la metodologia usada. Fa millorar l'enteniment de *Scrum* a través de la teoria, pràctiques, normes i altres valors de *Scrum*.
- ***Development Team:*** és l'equip que s'ocupa de desenvolupar les tasques pactades en el *backlog*. L'equip ha de ser capaç d'organitzar-se i gestionar el treball.

Cal remarcar l'existència dels "*Scrum events*", esdeveniments utilitzats en *Scrum* per a regular les reunions.

- ***Sprint:*** és una iteració de temps, de duració màxima d'un mes, on es proposa una feina per a l'equip i aquest ha de completar-la, fent d'ella un producte usable.
- ***Sprint Planning:*** aquest esdeveniment consisteix en la selecció de la feina que s'ha de fer durant un *Sprint*. En *sprints* d'un mes, té una duració màxima de vuit hores. Quant menor sigui el *sprint*, menor la duració. El "*Scrum Master*"s'ha d'ocupar de que tots els integrants del "*Development team*"estiguin presents i entenguin el concepte de la reunió.
- ***Daily Scrum:*** és una reunió diària d'uns 15 minuts com a màxim on cada integrant del *Scrum team* exposa davant de la resta de l'equip la feina que ha realitzat durant el dia. En cas de fer la reunió al principi del dia, es parla sobre el dia anterior. Serveix per a sincronitzar les activitats i planificar, per damunt, les 24 hores següents.
- ***Sprint Review:*** reunió al final del *Sprint* on es valora tota la feina realitzada, així com l'increment/decrement del *backlog*, o l'adaptació d'aquest últim en cas d'ésser necessari.
- ***Sprint Retrospective:*** parteix després del "*Sprint Review*"on l'equip analitza els punts forts i dèbils del *sprint* per a poder millorar en el següent *sprint*.

Si es volen saber les avantatges i/o inconvenients d'utilitzar aquesta metodologia anar a la subsecció **15.2** de l'annex.

[9] és la feina que l'equip ha d'anar fent per a completar el projecte.

9 Objectius del treball final de grau

Seguidament parlarem sobre els objectius tant generals com específics del projecte a realitzar.

9.1 Objectius Generals

L'objectiu principal d'aquest projecte és el de desenvolupar una aplicació web per a la gestió de cues de telèfons per al departament de Contact de l'empresa Ilerna Online.

9.2 Objectius Específics

Els objectius secundaris o específics sobre aquest projecte són els següents:

- Definir els requeriments del projecte.
- Realitzar la valoració i planificació del projecte.
- Analitzar i dissenyar la solució.
- Implementació de la solució.
- Tests per a cada implementació i ajudar en el desplegament de l'aplicació al servidor Devel-Core propi de l'empresa Ilerna Online.

10 Estat de l'Art

En aquest apartat parlarem sobre l'empresa Ilerna Online per tal d'informar i familiaritzar al lector sobre aquesta empresa, tal i com hem mencionat anteriorment.

10.1 Història d'Ilerna Online

Al 1954, a Lleida, es va fundar l'Acadèmia Tècnica on es van dur a terme les classes de repàs, mecanografia i informàtica. Al 1985 la Acadèmia Tècnica es va centrar en realitzar FP-I i FP-II.

Entre els anys 1992-1993 va nèixer Les Heures on s'impartien classes de ESO, Batxillerat i FP LOGSE, una formació que tenia una duració d'un any. Al 2006 es va originar el FP LOE la qual cosa va comportar que aquests estudis tinguessin una duració de dos anys. Dos anys després, al 2008, van recuperar la Acadèmia Tècnica, impulsant la FP ocupacional i continua.

Més tard, al 2012, es va fundar Ilerna absorbint les FP. A partir d'aquest any, Les Heures van deixar de donar FP i desde llavors van centrar la seva activitat en impartir ESO i Batxillerat. Així doncs, Ilerna va nèixer com un projecte global de dos grans marques dedicades a la formació professional amb més de 50 anys d'història. Ilerna representa la integració de totes les FP de Les Heres i l'Acadèmia Tècnica.

Al febrer de l'any 2014, es va fundar Ilerna Online com a centre d'estudis de formació professional a distància i oficial. Els fundadors varen ser Jordi Giné i Virginia Agelet. Dos anys més tard, al maig del 2016, va nèixer Ilerna Madrid i al juliol del 2017 Ilerna Italia.

10.2 Ilerna Online en l'Actualitat.

El Centre Integral de Formació Professional Ilerna imparteix cicles formatius, tant en la modalitat presencial a través d'Ilerna Lleida com en modalitat a distància a través d'Ilerna Online.

Ilerna centra tots els seus esforços en promoure una Formació Professional professionalitzadora i integradora, reforçant els valors tradicionals que sempre els ha identificat com són la cultura de l'esforç, l'atenció personalitzada, la internacionalització, les noves tecnologies i, sobre tot, la innovació constant en tots els àmbits, per tal d'aconseguir un gran repte i convertir-lo en una realitat immediata.

Les titulacions són 100% oficials i s'obtenen directament a través dels seus exàmens, sense haver de passar per cap tràmit més.

El perfil dels seus alumnes comprèn les edats de 25 a 45 anys que solen treballar, i/o tenen fills i/o fa temps que no estudien. Per aquest motiu, és important l'atenció i l'assessorament en cada moment, tant abans com després de realitzar la matrícula.

Cal destacar també que la matrícula no va per cicles formatius sinó per assignatures. Els/les alumnes poden matricular-se de les matèries que vulguin decidint què, quan i quant estudiar.

El servei que ofereixen com a centre d'estudis és impartir diferents modalitats de FP. Actualment, compten amb els següents cicles:

- CFGM Auxiliar d'Infermeria.
- CFGM Sistemes Microinformàtics i reds.
- CFGM Gestió Administrativa.
- CFGM Electromecànica de vehicles automòbils.
- CFGM Farmàcia i Parafarmàcia.
- CFGS Higiene Bucodental.
- CFGS Educació Infantil.
- CFGS Desenvolupament d'Aplicacions Multiplataforma.
- CFGS Desenvolupament d'Aplicacions Web.
- CFGS Animacions 3D, Jocs i entorns interactius.
- CFGS Màrqueting i Publicitat.
- CFGS Administració i Finances.
- CFGS Assistència a la Direcció.
- CFGS Dietètica.

11 Desenvolupament del Projecte

11.1 Iteració 1

11.1.1 Definició dels requeriments

El procés de reunió de requisits s'intensifica i es centra especialment en el *software*. Dintre del procés d'anàlisis és fonamental que a través d'una col·lecció de requeriments funcionals i no funcionals, el desenvolupador o desenvolupadors del *software* comprenguin completament la naturalesa dels programes que s'han de construir per desenvolupar l'aplicació, la funció requerida, comportament, rendiment i interconnexió. En el cas d'aquest projecte, el procés d'anàlisis i d'obtenció dels requeriments s'ha dut a terme a través del treball conjunt amb la Judit del departament de Contact d'Ilerna Online, la qual ens proporciona els paràmetres (peticions) que ens serveixen per realitzar l'aplicació per poder, d'aquesta manera, complir amb els objectius d'aquest projecte. En aquest cas, després de parlar amb la Judit he arribat a dur a terme el següent llistat estructurat i refinat de les funcionalitats del sistema, degudament classificades.

11.1.1.1 Requeriments Funcionals

Els requeriments funcionals són els que s'encarreguen de definir el que l'eina de *software* ha d'utilitzar. Defineix l'abast del sistema pel que fa a les accions que ha de realitzar, i pel que fa a la transferència de dades entre totes les diferents funcions del sistema.

En el cas d'aquest projecte, els principals requeriments funcionals són els següents:

- **R.1. - Requeriments funcionals bàsics.**
  - **R.1.1 -** El sistema haurà de mostrar un menú amb els següents camps: Agents, *Blacklist*, Cursos, Cues, Assignacions.
  - **R.1.2 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó Agent del menú, tant els noms com els cognoms dels agents i la cua a la qual pertanyen. A més, en la nova pantalla, hi ha d'haver alguna opció per tal de poder crear, modificar i eliminar un agent. També hi ha d'haver alguna opció per a que cada agent pugui exportar un excel.
  - **R.1.3 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó *Blacklist* del menú, tant els noms com els cognoms dels agents i dels alumnes al qual estan relacionats. Tot seguit, en la nova pantalla, hi ha d'haver alguna opció per poder eliminar i afegir una *blacklist*.
  - **R.1.4 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó Cursos del menú, tant els noms dels cursos com els noms de la família a la qual pertanyen. Tot seguit, en la nova pantalla, hi ha d'haver alguna opció per tal de poder modificar la família dels cursos. A més, aquesta nova pantalla ha de tenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
  - **R.1.5 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó Cues del menú, tots els noms de les cues. Tot seguit, en la nova pantalla, hi ha d'haver alguna opció per poder crear una nova cua.

- **R.1.6 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó Assignacions del menú, una taula amb el nom dels agents i amb els següents criteris: el número total d'estudiants que té cada agent amb la mitjana d'estudiants que haurien de tenir com a màxim. El número d'estudiants que té cada agent incloent els agents especials [10], per FCT, família del cicle (Sanitat, Informàtica, etc), i la repartició d'alumnes per agent ha de ser proporcional pel tipus de criteri. A més, es disposarà d'un llistat sense filtrar inicialment on es veuran tots els alumnes registrats a la seva cartera. Disposarà d'uns filtres concrets per poder facilitar la cerca (Alumne, FCT, Curs, Família, Assignacions i Agent).

- **R.2. - Requeriments relacionats amb el seleccionament d'opcions de la nova pantalla d'agents.**
  - **R.2.1 -** El sistema haurà de permetre la creació d'un nou agent en una nova pantalla, un cop seleccionada l'opció de crear agent, amb els següents camps: nom, cognoms, correu electrònic, cua a la qual s'associarà aquest agent, absències previstes (dia, mes, any, hora i una nota per tal de saber el motiu de l'absència). A més, aquesta nova pantalla ha de tenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
  - **R.2.2 -** El sistema haurà de permetre la modificació d'un agent existent en una nova pantalla, un cop seleccionada l'opció de modificar agent, amb els següents camps: nom, cognoms, correu electrònic, cua a la qual s'associarà aquest agent, absències previstes (dia, mes, any, hora i una nota per tal de saber el motiu de l'absència). A més, aquesta nova pantalla ha de tenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
  - **R.2.3 -** El sistema haurà de permetre l'eliminació d'un agent existent, un cop seleccionada l'opció d'eliminació d'agent, sempre que no tingui assignacions fetes.
  - **R.2.4 -** El sistema haurà de permetre que cada agent pugui exportar un excel, un cop seleccionada l'opció, amb els següents resultats sobre l'alumne: id, nom, cognoms, correu electrònic personal, correu electrònic incorporat per l'empresa, número de telèfon, nom del semestre, nom del curs, nom de la família, si l'alumne correspon a FCT o no i per últim, si l'alumne correspon a *Premium* o no. A més, no s'haurà de mostrar tot en una fulla excel sinó que cada fulla representarà el nom del semestre i hauran d'estar ordenades de manera descendent pel nom del semestre.

- **R.3. - Requeriments relacionats amb el seleccionament d'opcions de la nova pantalla de blacklist.**
  - **R.3.1 -** El sistema haurà de permetre la creació d'una nova *blacklist* en una nova pantalla, un cop seleccionada l'opció de crear *blacklist*, amb els següents camps: nom, cognoms de l'agent (els quals han de ser seleccionables ja que tindrem pocs agents) i nom, cognoms de l'estudiant (els estudiants sortiran a mesura que es vagin escrivint per a que es pugui seleccionar l'alumne ja que hi han molts alumnes). A més, aquesta nova pantalla ha de contenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
  - **R.3.2 -** El sistema haurà de permetre l'eliminació d'una *blacklist* un cop seleccionada l'opció d'eliminació de *blacklist*.

[10] representen aquells agents que porten alumnes que són premium o que pertanyen a centres col·laboradors. Fins i tot alumnes que pertanyen a ambdues coses.

- **R.4. - Requeriments relacionats amb el seleccionament d'opcions de la nova pantalla de cues.**
  - **R.4.1 -** El sistema haurà de permetre la creació d'una nova cua en una nova pantalla, un cop seleccionada l'opció de crear cues, amb el camp nom de la cua. A més, aquesta nova pantalla ha de tenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
- **R.5. - Requeriments relacionats amb la distribució dels alumnes entre els agents.**
  - **R.5.1 -** El sistema haurà de permetre l'assignació d'alumnes als agents tenint en compte els següents criteris:
    * Hi hauran dos tipus d'agents: agents especials i agents. Els agents especials s'encarregaran dels alumnes que són *premium* i/o pertanyen de centres col·laboradors. Els agents s'encarregaran de la resta d'alumnes que no pertanyen ni a premium ni a centres col·laboradors.
    * Per distribuir els alumnes entre els agents normals s'ha de considerar el següents criteris amb un ordre de priotitat:
      · El criteri amb més priotitat: el nombre d'alumnes que estan cursant FCT que tenen els agents. L'agent amb el menor nombre d'alumnes serà al que se li assignarà l'alumne. En cas d'empat entre agents es passa al segon criteri amb més prioritat.
      · El segon criteri amb més prioritat: el nombre total d'alumnes que tenen els agents. L'agent amb el menor nombre total d'alumnes serà al que se li assignarà l'alumne. En cas d'empat entre agents es passa a l'últim criteri amb més prioritat.
      · Últim criteri: el nombre d'alumnes, que tenen els agents, de la família del cicle al qual pertany l'alumne. L'agent amb el menor nombre d'alumnes de la família del cicle al qual pertany l'alumne serà al que se li assignarà aquest alumne. En cas d'empat entre agents s'haurà d'escollir aleatòriament entre els que han empatat.

11.1.1.2 Requeriments No Funcionals

Els requeriments no funcionals són aquells que defineixen el que l'eina de *software* ha de tenir pel que fa a l'aparença, sensació, operabilitat i manteniment.

En el cas d'aquest projecte, els principals requeriments no funcionals són els següents:

- **1. - Producte**
  - **1.1. - Eficiència**
    - **1.1.1. -** El sistema farà ús d'un algorisme de cerca per tal d'oferir un seleccionable amb tots els agents quan s'estigui creant una nova *blacklist*. El seleccionable no pot tardar més de 20 segons en sortir.
    - **1.1.2. -** El sistema farà ús d'un algorisme de cerca aproximada per tal d'oferir mentre s'estigui escrivint el nom i els cognoms d'un alumne. Tant el nom com els cognoms no poden tardar més de 30 segons en sortir.
  - **1.2. - Portabilitat**
    - **1.1.1. -** El component portal web ha de suportar la intercompatibilitat amb els diversos navegadors.
- **2. - Organització**
  - **2.1. - Desenvolupament**
    - **2.1.1 -** Llenguatge de programació PHP versió 7.2.
    - **2.1.2 -** Bases de dades MySQL versió 5.7.
    - **2.1.3 -** *Framework* Laravel Eloquent versió 5.8.
    - **2.1.4 -** Biblioteca PhpSpreadSheet.
    - **2.1.5 -** Servidor utilitzat devel-core (propi de Ilerna Online).

11.1.2 Planificació

En aquesta part de la primera iteració, és on esmentarem la planificació del projecte per començar a donar una estructura al projecte. Com bé hem comentat anteriorment, és la part més important ja que les següents fases partiran sobre aquesta.

Pel que fa a la planificació del projecte vam decidir que la part de ***backend*** la realitzaríem en Marc Olivé i jo amb el llenguatge de programació PHP (usant com a ***PHP IDE*** PhpStorm per aquest llenguatge), utilitzant com a *framework* **Laravel Eloquent** i la base de dades de ***MySQL***. En canvi, la part de ***frontend*** la farà l'Ismael López Pérez. Hem utilitzat aquestes eines ja que són les eines que s'usen a la pròpia empresa.

Per una banda, segons **HRCS** (*Human Resources Consulting Services*) notícies de llocs de treball, PHP és un dels llenguatges de programació més utilitzat per al desenvolupament *backend* dels projectes.

Per altra banda, escollir el *framework* adequat per a les nostres aplicacions és un dels aspectes més difícils de les etapes inicials del desenvolupament dels projectes. Tot i que els criteris generals per a un equip de desenvolupament per escollir el *framework* acostumen a ser: el cost de desenvolupament, la seva experiència amb ell, la popularitat del *framework*, etc. Hi ha diversos altres factors, com ara integracions de tercers, desplegament, proves i per tant, es necessita una consideració deliberada.
El llenguatge de script PHP té una varietat de *frameworks* amb robustes capacitats tècniques com: Laravel, Symfony, CodeIgniter, Yii 2, Phalcon, CakePHP, Zend, Slim, etc. No obstant això, Laravel ha mantingut la primera posició a la llista dels millors ***MVC*** de ***PHP***. En el cas d'aquesta empresa, la experiència tant amb el *framework* Laravel com en l'ús de les altres eines és el motiu principal del seu ús. A més, actualment, segons **Clarion Blog** ens explica 10 raons de perquè el *framework* Laravel és el millor PHP *framework* per utilitzar aquest 2019.

Seguidament, esmentarem resumidament les 10 raons les quals ens explica:

- **Suport MVC i enfocament orientat a objectes:** el primer i el millor avantatge d'utilitzar el *framework* de Laravel és que segueix: el patró d'arquitectura basat en model, vista i controlador i que té una bella sintaxi expressiva que el fa orientat a objectes.
- **Autenticació i autorització integrades:** Laravel ens proporciona una configuració inèdita per al sistema d'autenticació i autorització. És a dir, en només uns pocs comandaments artesanals, la nostra aplicació estarà equipada amb una autenticació i una autorització segures.
- **Sistema d'embalatge:** un sistema d'empaquetament tracta els programes de suport múltiples o les biblioteques que ajuden l'aplicació web a automatitzar el procés. Laravel utilitza un compositor com a gestor de dependències, que gestiona tota la informació necessària per gestionar els paquets. Els paquets són una manera excel·lent d'accelerar el desenvolupament i de proporcionar la funcionalitat que necessitem.
- **Sistema de fitxers múltiples:** Laravel també compta amb un suport integrat per al sistema d'emmagatzematge en núvol com Amazon S3 i Rack space Cloud Storage i, per descomptat, per a l'emmagatzematge local. És increïblement senzill canviar entre aquestes opcions d'emmagatzematge, ja que l'***API*** segueix sent el mateix per a cada sistema. Es poden utilitzar els tres sistemes en una sola aplicació per servir fitxers de múltiples ubicacions com en un entorn distribuït.
- **Consola Artesana:** Laravel té la seva pròpia interfície de línia d'ordres anomenada Artisan. Els usos més habituals d'Artisan inclouen la publicació de paquets, la gestió de migracions de bases de dades i la generació de codi per a nous controladors, models i migracions. Aquesta característica allibera el desenvolupador de la creació d'esquelet de codi adequat. Es pot ampliar la funcionalitat i les capacitats d'Artisan mitjançant l'implementació de noves ordres personalitzades.

- **Eloquent ORM:** és la implementació ORM integrada de Laravel. Laravel té el millor mapejador relacional d'objectes en comparació amb els altres frameworks que hi ha. Aquest mapatge relacional d'objectes ens permet interactuar amb els nostres objectes de base de dades i les nostres relacions de base de dades mitjançant la sintaxi expressiva.

- **Motor de plantilla:** Laravel ve amb el motor de plantilla incorporat conegut com a *Blade Template Engine*. *Blade Template Engine* combina un o més plantilles amb un model de dades per produir vistes resultants, fent-ho mitjançant la transpiració de les plantilles en un codi PHP en memòria cau per millorar el rendiment. *Blade* també proporciona un conjunt de les seves pròpies estructures de control, com ara declaracions i bucles condicionals, que es mapen internament als seus homòlegs PHP.

- **Programació de tasques:** A la versió de Laravel 5.0 es va afegir a la línia de comandaments de l'Artisan utilitats que ens permeten la programació programàtica de tasques executades periòdicament.

- **Esdeveniments i difusió:** Laravel té un concepte anomenat broadcasting que és útil en l'aplicació web moderna per implementar dades en temps real. La difusió li permet compartir el mateix nom d'esdeveniment entre el servidor i el client, de manera que podrà arrossegar dades en temps real de l'aplicació.

- **Testing:** Quan es tracta de provar l'aplicació, Laravel proporciona per defecte la prova d'unitat per a l'aplicació, que conté proves que detecten i eviten les regressions en el *framework*. La integració de la unitat PHP, com ara un *framework* de proves, és molt fàcil en l'aplicació de Laravel. A més d'aquestes unitats, les proves es poden executar a través de l'eina de comandament artesanal proporcionada. A més d'aquestes característiques, Laravel també té paquets oficials que són útils quan integren diferents característiques de l'aplicació.

Un cop sabíem quines eines utilitzar, vàrem començar a descarregar-les i vam instal·lar una llibreria externa anomenada *PhpSpreedSheet* per tal de poder dur a terme l'excel que ens demanaven a un dels requeriments esmentats anteriorment.

11.1.3 Disseny

El disseny de *software* és realment un procés de molts passos però que es classifiquen dintre d'un mateix. En general, l'activitat del disseny es refereix a l'establiment de les estructures de dades, l'arquitectura general del *software*, representacions d'interfícies i algorismes.

En aquesta part de l'iteració, un cop acabada la part de planificació, esmentarem les decisions de disseny referents al projecte en qüestió i introduirem els diagrames UML inicials que ens van ajudar per començar a tenir una mera idea per tal de trobar una solució.

Per a representar estructures de dades simples i llistes associatives, anomenades objectes hem utilitzat el format ***JSON***. Aquest format està constituït per dos estructures:

- Una col·lecció de parells de nom/valor. En diversos llenguatges això és conegut com un objecte, registre, estructura, diccionari, tabla hash o llista de claus.

- Una llista ordenada de valors. En la majoria dels llenguatges, això s'implementa com vectors, llistes o seqüències.

Majoritàriament, hem utilitzat llistes, vectors i col·leccions de dades ja que al tenir un conjunt de dades ens anava millor treballar-hi.

Pel que fa a l'arquitectura general del *software* hem utilitzat el patró Model-Vista-Controlador (MVC) el qual està explicat a l'annex. Encara que es puguin trobar diferents implementacions de MVC, el flux de control que segueix generalment és el següent:

- L'usuari interactua amb la interfície d'usuari d'alguna manera (per exemple l'usuari pitja un botó, enllaç, etc.).
- El controlador rep (per part dels objectes de la interfície-vista) la notificació de l'acció sol·licitada per l'usuari. El controlador gestiona l'event que arriba, freqüentment a través d'un gestor d'events (*handler*) o *callback*.
- El controlador accedeix al model, actualitzant-lo, possiblement modificant-lo de manera adequada a l'acció sol·licitada per l'usuari (per exemple, el controlador actualitza el carro de la compra de l'usuari.) Els controladors complexes estan sovint estructurats usant un patró de comanda que encapsula les accions i simplifica la seva extensió.
- El controlador delega als objectes de la vista la tasca de desplegar la interfície d'usuari. La vista obté les seves dades del model per a generar la interfície apropiada als objectes per a l'usuari on es reflecteixen els canvis en el model (per exemple, produeix un llistat del contingut del carro de la compra). El model no ha de tenir coneixement directe sobre la vista.
- La interfície d'usuari espera noves interaccions de l'usuari, començant el cicle novament.

Encara que originalment MVC va ser desenvolupat per aplicacions d'escriptori, ha sigut àmpliament adaptat com arquitectura per a dissenyar i implementar aplicacions web en els principals llenguatges de programació. S'han desenvolupat multitud de *frameworks*, comercials i no comercials, que implementen aquest patró. Com bé hem esmentat anteriorment, el *framework* utilitzat per part nostra ha sigut Laravel Eloquent i l'hem utilitzat perquè els professionals del departament d'IT d'aquesta empresa estan acostumats i tenen experiència sobre ell.

Pel que fa als diagrames inicials UML que vam realitzar per a la base de dades són els següents:

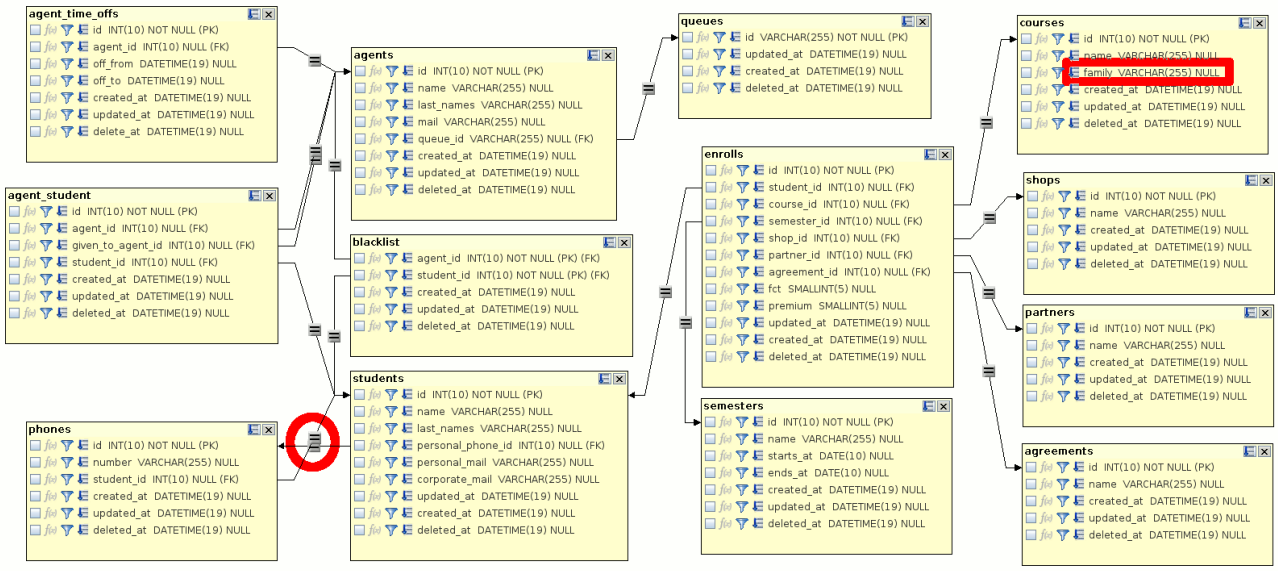

Figura 1: Primera versió del Diagrama UML sobre la base de dades de l'aplicació.

Aquest diagrama representa la primera idea general que vam pensar amb la intenció de complir tots els requeriments demanats anteriorment. La idea consistia en realitzar diverses taules a la base de dades relacionades entre elles per tal d'obtenir tots els camps que es demanaven i poder complir així els requeriments. Com bé observem en la figura 1 tenim una taula anomenada *agents* la qual representa els professionals que s'encarreguen de l'atenció al client o també anomenats agents amb els seus camps corresponents. Una taula anomenada *blacklist* que representa una llista on hi han els agents amb els seus alumnes que no tenen bona relació i viceversa. Seguidament, tenim una taula anomenada *students* la qual representa els alumnes amb la informació que necessitem sobre aquests.

Després, tenim la taula *queues* que representa les cues a les quals s'associen als agents. Tot seguit, disposem d'una taula anomenada *enrolls* la qual representa tota aquella informació sobre la matriculació dels alumnes que necessitem per tal de saber per exemple a quin semestre està matriculat, a quin curs, etc. A més, tenim dues taules anomenades *semesters* i *courses* les quals contenen informació sobre tots els semestres i cursos respectivament. També disposem d'una taula anomenada *agent_time_off* que ens serveix per gestionar quan un agent està de baixa o de vacances. Tanmateix, vam realitzar una taula anomenada *agent_student* la qual ens serveix per saber quins alumnes corresponen a cada agent. Per últim tenim una taula anomenada *phones* la qual conté informació sobre els telèfons dels estudiants.

En la figura 1 podem observar un cercle vermell a la relació de *students* a *phones* el qual representa un petit conflicte ja que per posar un alumne primer necessitaríem l'*id* del seu telèfon principal, i per posar el telèfon i tenir l'id del telèfon necessitaríem primer posar l'alumne. A més, també tenim encerclat en vermell el camp *family* de la taula *courses* ja que vam pensar després que en aquest camp hi tindríem noms repetits a la base de dades i el millor hagués sigut crear una taula nova anomenada *family* la qual tingués com a clau primària un *id* i utilitzar aquest *id* com a clau forana a la taula *courses* en comptes del camp *family*. D'aquesta manera no tindríem noms de *family* duplicats.

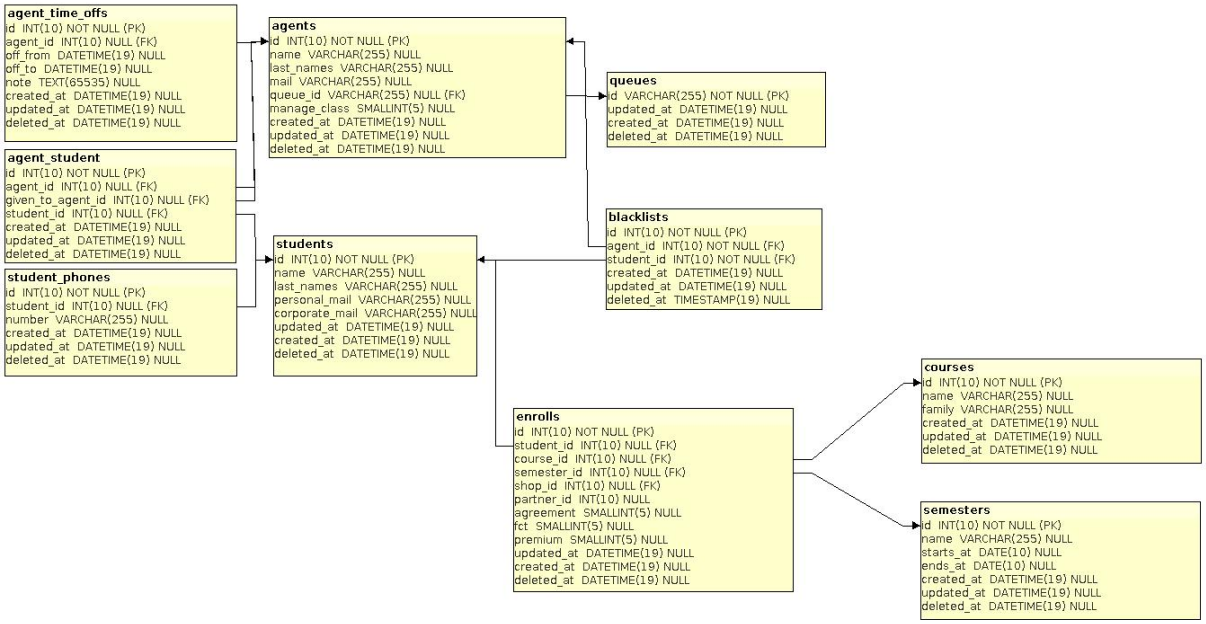

Figura 2: Segona versió del Diagrama UML sobre la base de dades de l'aplicació.

Més tard, vam realitzar un nou diagrama UML el qual representa la figura 2. Aquesta figura correspon a una modificació de la idea que teníem sobre el UML de la figura 1. Aquesta nova idea consistia en treure les taules *agreements, partners* i *shops* ja que vam veure que era innecessari tenir tres taules quan les podíem agrupar-ho a la taula *enrolls*. Ho vam pensar així ja que no feia falta crear la taula *shops*, en aquesta primera iteració, perquè no hi havien requeriments sobre aquesta però ens imaginàvem que més endavant si que l'hauríem de tenir en compte. Per tant, el que vam fer va ser deixar el camp *shop_id* dintre de la taula *enrolls*. Pel que fa a les taules *partners* i *agreements* les vam treure ja que en els requeriments sols s'especificava que hi hagués les opcions de saber si un alumne pertany a *partner* i/o a centres col·laboradors. Per tant, introduint un booleà dintre de la taula *enrolls* ja en teníem suficient per saber si l'alumne pertany a *partner* i/o a centres col·laboradors. Aquests camps els vàrem introduir a la taula *enrolls* ja que no deixa de ser informació sobre l'alumne. A més, la relació que vam encerclar en vermell , que anava de la taula *students* a *student_phones*, la vam eliminar (eliminant el *personal_phone_id*) ja que no necessitàvem diferenciar entre el telèfon principal i els secundaris de l'alumne pels requeriments demanats.

11.1.4 Implementació

En aquesta part de la primera iteració, un cop acabada la part de disseny, esmentarem la implementació [11] que hem dut a terme. És a dir, el desenvolupament dels Models, dels Controladors en classes. Esmentarem tant les funcions que conté cada model i controlador com les seves rutes corresponents. Finalment mostrarem els tests adients que han servit per una mera comprovació sobre la implementació. Començarem explicant les rutes, seguidament els models, després els controladors i finalment els tests sobre la implementació.

11.1.4.1 Routes

Totes les rutes del *framework laravel* es defineixen als fitxers de la ruta, que es troben al directori de rutes. Aquests fitxers es carreguen automàticament pel *framework*. Les rutes en el fitxer routes/api.php les hi assigna el grup de ***middleware*** api. El *middleware* proporciona un mecanisme convenient per filtrar les sol·licituds ***HTTP*** que entren a la nostra aplicació. Per exemple, Laravel inclou un *middleware* que verifica si l'usuari de la nostra aplicació està autenticat. Si l'usuari no està autenticat, el *middleware* redirigirà l'usuari a la pantalla d'inici de sessió. Tanmateix, si l'usuari està autenticat, el *middleware* permetrà que la sol·licitud continuï més en l'aplicació.
Les rutes de Laravel més senzilles consisteixen en una URI i una crida de retorn de tancament.
Com per exemple:

```
<?php
    Route::get('/', function()
    {
        return 'Hello World';
    })
?>
```

11.1.4.2 Api.php

En aquest fitxer és on es registren les rutes de l'API per a la nostra aplicació. Aquestes rutes es carreguen pel RouteServiceProvider dins d'un grup que se li assigna el grup *middleware* "api".
Les rutes de l'API que hem realitzat són les següents:

11.1.4.2.1 Rutes Students

```
<?php
    Route::middleware('auth:api')
    -> get('/students', function(Request $request)
    {
        return App\Http\Controllers\StudentController::
        readStudents($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i retorna la informació de tots els estudiants cridant la funció **readStudents()** del controlador StudentController.

```
<?php
    Route::middleware('auth:api')
    -> get('/students/propose_assignments/{semester_id}', function($semesterID)
    {
        return App\Http\Controllers\StudentController::
        getProposedAssignments($semesterID);
    });
?>
```

[11] traduir el disseny en una forma legible per la màquina.

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre l'assignació proposada d'*agent-student* cridant la funció **getProposedAssignments()** del controlador StudentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/students/agents', function()
    {
        return App\Student::with(['agents', 'blacklist'])
        -> has('agents')-> get();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre els alumnes que tenen un agent, ens retorna per cada alumne el seu agent i si pertany a la *blacklist* o no fent ús del model Student i utilitzant les relacions adients a aquest model.
Un exemple de JSON retornat seria el següent:


```
[{"id":1,"name":"Marc","last_names":"Apellido1 Apellido2","personal_phone_id":605236598,
"personal_mail":"marc@examplel.com","corporate_mail":"marc@examplecorporate.com",
"agents":[{"id":1}],"blacklist":[{"id":2,"agent_id":2,"student_id":1}]}
```

```php
<?php
    Route::middleware('auth:api')
    -> get('/students/full', function()
    {
        return App\Student::
        with(['agents', 'blacklist', 'studentPhones', 'enrolls'])
        -> get();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre: els alumnes, incloent els seus números de telèfon, el seu agent, si pertany a la *blacklist* o no. Per últim també es retorna informació sobre les matrícules dels alumnes fent ús del model Student i utilitzant les relacions adients a aquest model.
Un exemple de JSON retornat seria el següent:


```
[{"id":1,"name":"Pau","last_names":"Apellido1 Apellido2","personal_phone_id":605236598,
"personal_mail":"pau@example.com","corporate_mail":"pau@examplecorporate.com",
"agents":[{"id":1}],"blacklist":[{"id":2,"agent_id":2,"student_id":1}],
"student_phones":[], "enrolls":[{"id":1,"course_id":1,"semester_id":1,
"shop_id":1,"agreement":null,"fct":0,"premium":1}]}]
```


```php
<?php
   Route::get('/students/queue', function(Request $request)
   {
        $params = $request->all();
        if(!array_key_exists('api_token', $params) ||
        $params['api_token']!='F8oYA8RbE9kij1uUdqxdK6AfuMVBmtT96BsosHwkQ5v3i7t3rd')
            return;
        return App\Http\Controllers\StudentController::
        getQueueFromPhone($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre la cua a la qual pertany un estudiant per mitjà del seu número de telèfon, fent ús de la funció **getQueueFromPhone()** del controlador StudentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/students/search/{name}', function($name)
    {
        return App\Http\Controllers\StudentController::
        findByName($name);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre els estudiants (nom i cognoms), fent ús de la funció **findByName()** del controlador StudentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/students/{id}', function($id)
    {
        return App\Http\Controllers\StudentController::read($id);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre l'estudiant. Tot això fent ús del seu identificador (id) passat per paràmetre en la URL i utilitzant la funció **read()** del controlador StudentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/students/{id}/agents', function($id)
    {
        return App\Student::where('id', $id)
        -> with(['agents', 'blacklist'])
        -> get();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre: l'alumne mitjançant el id que hem rebut com a paràmetre, el seu agent i si pertany a la *blacklist*. Usant el model Student i utilitzant les relacions adients a aquest model. Un exemple de JSON retornat seria el següent:


```
[{"id":3,"name":"Eloi","last_names":"Apellido1 Apellido2","personal_phone_id": 605236547,
"personal_mail":"eloi@example.com","corporate_mail":"eloi@examplecorporate.com",
"agents":[{"id":3}],"blacklist":[{"id":4,"agent_id":4,"student_id":1}]}
```


```php
<?php
    Route::middleware('auth:api')
     -> get('/students/{id}/full', function($id)
     {
        return App\Student::where('id', $id)
        -> with(['agents', 'studentPhones', 'blacklist', 'enrolls'])
        -> get();
     });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre: l'alumne mitjançant l'id que hem rebut com a paràmetre, incloent els seus números de telèfon, el seu agent i si pertany a la *blacklist* o no. Per últim també es retorna informació sobre les matrícules dels alumnes fent ús del model Student i utilitzant les relacions corresponents a aquest model.
Un exemple de JSON retornat seria el següent:


```
[{"id":7,"name":"Josep","last_names":"Apellido1 Apellido2","personal_phone_id":601253558,
"personal_mail":"josep@example.com","corporate_mail":"josep@examplecorporate.com",
"agents":[{"id":1}],"blacklist":[{"id":2,"agent_id":2,"student_id":7}],
"student_phones":[], "enrolls":[{"id":1,"course_id":1,"semester_id":1,
"shop_id":1,"agreement":null,"fct":0,"premium":1}]}]
```

11.1.4.2.2 Rutes Agents

```php
<?php
    Route::middleware('auth:api')
    -> get('/agents', function()
    {
        return App\Http\Controllers\AgentController::
        readAll();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre tots els agents (*id, name, last_names, mail, manage_class, queue i agent_time_off*). Tot això fent ús de la funció **readAll()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/agents/full', function()
    {
        return App\Http\Controllers\AgentController::
        readAllFull();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre tots els agents (*id, name, last_names, mail, manage_class, queue, agent_time_off i blacklist*). Amb l'ajuda de la funció **readAllFull()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/agents/full/{id}', function($id)
    {
        return App\Http\Controllers\AgentController::
        readFull($id);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre l'agent (*id, name, last_names, mail, manage_class, queue, agent_time_off i blacklist*) mitjançant l'id que hem rebut com a paràmetre. Usant la funció **readFull()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/agents/{id}/excel', function($id)
    {
        return App\Http\Controllers\AgentController::
        agentExcel($id);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar un excel mitjançant l'id de l'agent que hem rebut com a paràmetre. Fent ús de la funció **agentExcel()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> get('/agents/{id}', function($id)
    {
        return App\Http\Controllers\AgentController::
        read($id);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar la informació sobre l'agent (*id, name, last_names, mail, manage_class, queue, agent_time_off*) mitjançant l'id de l'agent que hem rebut com a paràmetre. Amb l'ajuda de la funció **read()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> post('/agents', function(Request $request)
    {
        return App\Http\Controllers\AgentController::
        create($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus POST i de retornar la informació sobre l'agent creat i dels que ja teníem creats (*id, name, last_names, mail, manage_class, queue, agent_time_off*). Fent ús de la funció **create()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> put('/agents', function(Request $request)
    {
        return App\Http\Controllers\AgentController::
        update($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus PUT i de retornar la informació sobre l'agent modificat i dels que ja teníem creats (*id, name, last_names, mail, manage_class, queue, agent_time_off*). Usant la funció **update()** del controlador AgentController.

```php
<?php
    Route::middleware('auth:api')
    -> delete('/agents/{id}', function($id)
    {
        return App\Http\Controllers\AgentController::
        delete($id);
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus DELETE i de retornar la informació sobre els agents (*id, name, last_names, mail, manage_class, queue, agent_time_off*) exceptuant l'agent que hem eliminat fent ús de la funció **delete()** del controlador AgentController.

11.1.4.2.3 Rutes Queues

```php
<?php
    Route::middleware('auth:api')
    -> get('/queues', function()
    {
        return App\Queue::all();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar l'id de totes les cues fent ús del model Queue i cridant el mètode all() del *framework laravel*.

```php
<?php
    Route::middleware('auth:api')
    -> post('/queues', function(Request $request)
    {
        return \App\Http\Controllers\QueueController::
        create($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus POST i de retornar l'id de totes les cues i l'id que hem creat fent ús de la funció **create()** del controlador QueueController.

11.1.4.2.4 Ruta AgentStudent

```php
<?php
    Route::middleware('auth:api')
    -> post('/agents_students', function(Request $request)
    {
        return App\Http\Controllers\AgentStudentController::
        insertUpdateMany($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus POST i de retornar tots el agent_id i el student_id incloent els nous creats fent ús de la funció **insertUpdateMany()** del controlador AgentStudentController.

11.1.4.2.5 Rutes Cursos

```php
<?php
    Route::middleware('auth:api')
    -> get('/courses', function()
    {
        return App\Course::all();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar els camps: *id, name, family* usant el model *Courses* i de la funció all() del *framework Laravel*.

```php
<?php
    Route::middleware('auth:api')
    -> put('/courses', function(Request $request)
    {
        $controller = new App\Http\Controllers\CourseController();
        return $controller->update($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus PUT, modificar i retornar els camps *id, name, family* utilitzant la funció **update()** del controlador CoursesController.

11.1.4.2.6 Rutes Blacklist

```php
<?php
    Route::middleware('auth:api')
    -> get('/blacklist', function()
    {
        return App\Blacklist::all();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i retornar els camps: *id, agent_id, student_id* fent ús de la funció all() del *framework* laravel. Utilitzant així el model *Blacklist*.

```php
<?php
    Route::middleware('auth:api')
    -> post('/blacklist', function(Request $request)
    {
        $controller = new App\Http\Controllers\BlacklistController();
        return $controller->create($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus POST i retornar els camps: *id, agent_id, student_id* incloïent els nous creats fent ús de la funció **create()** del controlador Blacklist-Controller.

```php
<?php
    Route::middleware('auth:api')
    -> delete('/blacklist', function(Request $request)
    {
        $controller = new App\Http\Controllers\BlacklistController();
        return $controller->delete($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus DELETE i retornar els camps: *id, agent_id, student_id* exceptuant els eliminats fent ús de la funció **delete()** del controlador BlacklistController.

11.1.4.2.7 Rutes Semester

```php
<?php
    Route::middleware('auth:api')
    -> get('/semesters', function()
    {
        return App\Semester::all();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i retornar els camps: *id, name, starts_at, ends_at* . Usant la funció all() del *framework* Laravel i utilitzant així el model *Semester*.

11.1.4.2.8 Rutes Token

```php
<?php
    Route::middleware('auth:api')
    -> get('/token_test', function(Request $request)
    {
        return "OK";
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i de retornar un "OK" si el token existeix a la base de dades.

11.1.4.3 Models

Totes les següents classes estenen de la classe Model exceptuant les classes Template.php i User.php.

11.1.4.3.1 AgentStudent.php.

Aquesta classe conté variables de tipus *protected* **$hidden** i **$fillable**, encarregades de filtrar el contingut que desitgem. També conté dues funcions públiques anomenades student() i agent() les quals s'encarreguen de vincular aquesta classe amb altres classes del model de manera que la relació *belongsTo* definida dintre d'aquestes dues funcions ens permet definir un model per defecte que es retornarà si la relació donada és nul·la. Aquest patró sovint es coneix com a patró d'objectes nul i pot ajudar a eliminar els controls condicionals del nostre codi. De manera que un AgentStudent pertany a un *agent* i a un *student*.

11.1.4.3.2 Agent.php

Aquesta classe conté variables de tipus *protected* **$hidden**, **$fillable**, encarregades de filtrar el contingut que volem. A més, conté quatre funcions les quals s'encarreguen de relacionar aquesta classe amb altres models:

- **students():** la qual dur a terme la relació ***ManyToMany***. Ex: Un Agent pot tenir molts students.
- **blacklist():** la qual estableix la relació ***OneToMany***. Ex: Un Agent pot estar més d'una vegada a una blacklist.
- **queue():** la qual defineix la relació ***OneToOne***. Ex: Un Agent pertany a una Queue.
- **agentTimeOff():** la qual estableix la relació ***OneToMany***.Ex: Un Agent pot tenir més d'un *time off* (Temps de baixa o vacances).

11.1.4.3.3 AgentTimeOff.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat. A més, conté la funció agents() la qual s'encarrega de dur a terme la relació ***ManyToMany***. Ex: un *AgentTimeOff* pot pertànyer a més d'un *Agent*.

11.1.4.3.4 Blacklist.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat. A més, conté les següents funcions:

- **student():** la qual s'encarrega de dur a terme la relació ***OneToMany***. Ex: una *blacklist* pertany a molts *students*.
- **agent():** la qual s'encarrega de la relació ***OneToMany***. Ex: una *blacklist* pertany a molts *agents*.

11.1.4.3.5 Course.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat. A més, conté una altra variable anomenada **$incrementing** la qual la inicialitzem a fals per poder recollir més endavant l'id ja que a la Base de dades aquest id és auto incremental.

11.1.4.3.6 Enroll.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat. També conté les funcions:

- **student():** la qual defineix la relació ***OneToMany***. Ex: un enroll pertany a molts *Students*.
- **course():** la qual estableix la relació ***OneToMany***. Ex: un *enroll* pertany a molts *Course*.

11.1.4.3.7 Queue.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat. A més, conté altres variables com ara **$incrementing** la qual la inicialitzem a fals ja que no volem que el id sigui auto incremental i com que tampoc volem que sigui de tipus *integer* fem servir la variable **$keyType** inicialitzada com a *string*. També conté la funció agents() la qual defineix la relació ***OneToMany***.Ex: una *Queue* pot tenir molts *Agents*.

11.1.4.3.8 Semester.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per filtrar el contingut desitjat.

11.1.4.3.9 Student.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per el contingut desitjat. A més a més, conté les següents funcions:

- **studentPhones():** la qual s'encarrega de la relació ***OneToMany***. Ex: un *Student* pot tenir més d'un *StudentPhone*.
- **enrolls():** s'encarrega d'establir la relació ***OneToMany***. Ex: un *Student* pot tenir més d'un *Enroll*.
- **agents():** la qual defineix la relació ***ManyToMany***. Ex: un *Student* pot tenir molt *Agents*.
- **blacklists():** la qual s'encarrega de dur a terme la relació ***OneToMany***. Ex: un *Student* pot estar a més d'una *blacklist*.

11.1.4.3.10 StudentPhone.php

Aquesta classe igual que les anteriors també té dues variables **$hidden** per tal d'amagar el contingut que no volem i la variable **$incrementing** inicialitzada a fals per tal de que l'id deixi de ser auto incrementable. A més, tenim una funció anomenada student() la qual s'encarrega de l'establiment de la relació ***OneToMany***. Ex: un *StudentPhone* pertany a un *Student*.

11.1.4.3.11 Template.php

Aquesta classe igual que les anteriors conté la variable **$hidden** per tal d'amagar el contingut que no volem mostrar.

11.1.4.3.12 User.php

Aquesta classe estén de la classe Authenticatable i també té dues variables **$hidden** per tal d'amagar el contingut que no volem i la variable **$fillable** per mostrar el contingut que desitgem.

11.1.4.4 Controllers

Totes les següents classes estenen de la classe Controller.

11.1.4.4.1 AgentController.php

Aquesta classe s'encarrega de crear, mostrar, actualitzar, eliminar agents i de crear un excel per l'agent. Conté les següents funcions:

- **create():** comencem creant la transacció per la base de dades ja que aquesta aporta una fiabilitat superior a les bases de dades. Per exemple, si disposem d'una sèrie de consultes ***SQL*** que s'han d'executar en conjunt, amb l'ús de transaccions podem tenir la seguretat de que mai ens quedarem a mig camí de la seva execució. De fet, podríem, dir que les transaccions aporten una característica de "desfer"a les aplicacions de bases de dades. Una vegada creada la transacció omplim el *json* amb els camps *id, name, last_names, mail, manage_class, queue i agent_time_off* corresponent que ens passen. Un exemple del format que rebem seria:

  ```
  [{"id": 7, "name": "Agente", "last_names": "006",
  "mail": "agenten006@example.com", "manage_class": 0,
  "queue":{"C8766"}, "agent_time_off":[]}]
  ```

  Si tot ha anat bé fem el *commit* a la base de dades, sinó realitzem un *rollback* per desfer els canvis. Finalment retornem el JSON. Un exemple del format respost seria:

  ```
  [{"id":1,"name":"Agente001","last_names":"Entega001",
  "mail":"agententega001@example.com","manage_class":0,
  "queue":{"id":"C8741"},"agent_time_off":[]},
  {"id":2,"name":"Agente002","last_names":"Entega002",
  "mail":"agententega002@example.com","manage_class":0,
  "queue":{"id":"C8766"},"agent_time_off":[]}]
  ```

- **read():** retornem un JSON amb els següents camps sobre l'agent del qual hem buscat pel seu id: id, nom, cognoms, correu electrònic, si és estàndard o especial, és a dir si és premium o és de conveni, l'id de la cua al qual correspon aquest agent i per últim el seu *time off*.

  ```
  [{"id":3,"name":"Fernando","last_names":"Apellido1 Apellido2",
  "mail":"fernando@example.com","manage_class":1,
  "queue":{"id":"C300"},"agent_time_off":[]}]
  ```

- **readFull():** retornem un JSON amb els següents camps sobre l'agent del qual hem buscat pel seu id: id, nom, cognoms, correu electrònic, si és estàndard o especial, és a dir si són premium o són de conveni, l'id de la cua al qual correspon aquest agent, el seu *time off* i per últim si pertany a la *blacklist* o no. Per exemple:

  ```
  [{"id":2,"name":"Óscar","last_names":"Apellido1 Apellido2",
  "mail":"oscar@example.com","manage_class":1,
  "queue":{"id":"C200"},"agent_time_off":[],"blacklist":[]}]
  ```

- **readAll():** retornem un JSON amb els següents camps sobre tots els agents: id, nom, cognoms, correu electrònic, si és estàndard o especial (és a dir, si són premium o són de conveni), l'id de la cua al qual correspon i el seu *time off*. Per exemple:

  ```
  [{"id":1,"name":"David","last_names":"Apellido1 Apellido2",
  "mail":"david@example.com","manage_class":0,"queue":{"id":"C100"}
  ,"agent_time_off":[]}]
  ```

- **readAllFull():** retornem un JSON amb els següents camps sobre tots els agents: id, nom, cognoms, correu electrònic, si és estàndard o especial, és a dir si són premium o són de conveni, l'id de la cua al qual correspon cada agent, el seu *time off* i per últim si pertany a la *blacklist* o no. Per exemple:


```
[{"id":1,"name":"Pedro","last_names":"Apellido1 Apellido2",
"mail":"pedro@example.com","manage_class":0,"queue":{"id":"C100"},
"agent_time_off":[{"id":1,"off_from":"2018-02-04 00:00:00",
"off_to":"2018-02-24 00:00:00","note":null}],"blacklist":[]}]
```


- **update():** comencem creant la transacció per la base de dades. Una vegada creada la transacció omplim el *json* amb els camps *name, last_names, mail, queue i agent_time_off* corresponent que ens passen. Un exemple de format que rebem seria:


```
[{"id": 1, "name": "Agente001", "last_names": "Entega001",
"mail": "agentenentega001@example.com", "manage_class": 0,
"queue":{"C8767"}, "agent_time_off":[]},
{"id":6,"name":"Gerald","last_names":"Apellido1 Apellido2",
"mail":"gerald@mail.com", "manage_class":0,
"queue":{"id":"C8767"},"agent_time_off":[]}]
```


  Si tot ha anat bé realitzem la modificació i realitzem el *commit* a la base de dades, sinó realitzem un *rollback* per desfer els canvis. Finalment retornem el JSON. Un exemple de format de resposta seria:


```
[{"id":1,"name":"Agente001","last_names":"Entrega001",
"mail":"agenteentrega001@example.com","manage_class":0,
"queue":{"id":"C8767"},"agent_time_off":[]},
{"id":6,"name":"Maria","last_names":"Apellido1 Apellido2",
"mail":"maria@example.com","manage_class":0,"queue":{"id":"C8767"},
"agent_time_off":[]}]
```


- **delete():** comencem creant la transacció per la base de dades. Seguidament, obtenim els agents que no tenen estudiants. Mirem si existeix un agent que no tingui estudiants. Si no és així, realitzem un *rollback* perquè o bé no existeix aquest agent o si que existeix però té alumnes assignats i per tant no el podem eliminar. D'altra banda si tenim un agent que no té estudiants assignats marquem com a *soft delete* que aquest agent ja no existeix a les taules *blacklist*, *agentTimeOff* i a la taula *agent*. Si tot ha anat bé realitzem el *commit* per a guardar els canvis, sinó fem un *rollback* per tal de desfer els canvis. Finalment, retornem el JSON.

- **agentExcel():** primerament creem un excel fent ús de Spreadsheet(). Li posem l'estil de lletra que voldrem utilitzar i la mida. Seguidament, realitzem una consulta amb laravel amb l'id de l'agent que ens han passat per tal d'obtenir un JSON amb la següent informació sobre els estudiants d'aquest agent en concret: *id, name, last_names, personal_mail, corporate_mail, phones, semester, course, family, fct* i finalment si l'estudiant és premium o no. Seguidament, mirem si l'id de l'agent que ens han passat existeix o no. Si no existeix retornem un excel buit, d'altra banda si existeix retornem l'excel amb els camps esmentats anteriorment. Després, ordenem els camps del JSON per cognoms de manera ascendent. Tot seguit, cridem a la funció making_rows(). Més tard, com que sempre se'ns crea una pàgina buida al crear l'excel l'eliminem i realitzem una consulta amb Laravel per tal d'obtenir els semestres ordenats per starts_at (data de creació del semestre) de manera descendent. Seguidament, ordenem les pestanyes de l'excel pel nom del semestre. Finalment, redireccionem l'excel a l'output perquè quan l'usuari posi la url amb l'id de l'agent se li descarregui l'excel directament.

- **making_rows():** Aquesta funció s'encarrega d'anar omplint l'excel amb els camps *id, name, last_names, personal_mail, corporate_mail, phones, semester, course, family, fct* corresponents a cada estudiant d'aquell agent en concret. També s'encarrega de cridar a la funció initialize_sheet() la qual l'explicarem després. A més, també s'encarrega d'adaptar les files i les columnes de l'excel al text com a millora per a que l'agent ho pugui veure bé.

- **initialize_sheet():** Aquesta funció s'encarrega d'inicialitzar cada nova pàgina que creem de l'excel. La inicialització correspon en posar el nom i els cognoms de l'agent pel qual va dirigit aquest excel, una imatge de l'empresa. Finalment també s'encarrega d'establir i donar color a les capçaleres amb els camps corresponents a: *id, name, last_names, personal_mail, corporate_mail, phones, semester, course, family, fct.*

11.1.4.4.2 AgentStudentController.php

Aquesta classe s'encarrega d'actualitzar el contingut de la taula *agent_student* a la base de dades. Conté la següent funció:

- **insertUpdateMany()**: Comencem creant la transacció per la base de dades. Una vegada creada la transacció fem un soft delete[12] a la taula agentStudent. Seguidament, si tot ha anat bé realitzem un *insert* sobre la nova modificació i realitzem el *commit* a la base de dades, sinó realitzem un *rollback* per desfer els canvis. Finalment retornem el JSON amb els camps *agent_id, student_id.*

11.1.4.4.3 BlacklistController.php

Aquesta classe s'encarrega de crear i eliminar el contingut de la taula *blacklist* a la base de dades. Conté les següents funcions:

- **create():** Comencem creant la transacció per la base de dades. Una vegada creada la transacció omplim el json amb els camps *agent_id i student_id.* Un exemple del format que rebem seria:

  ```
  [{"id": 1, "agent_id": 1, "student_id": 16595},
  {"id": 4, "agent_id": 2, "student_id": 497}]
  ```

  Si tot ha anat bé fem el *commit* a la base de dades, sinó realitzem un *rollback* per desfer els canvis. Finalment retornem el JSON. Un exemple del format que retornem seria:

  ```
  [{"id":1,"agent_id":1,"student_id":16595},
  {"id":4,"agent_id":2,"student_id":497}]
  ```

- **delete():** Comencem creant la transacció per la base de dades. Una vegada creada la transacció comprovem que l'*agent_id, student_id* passats siguin els corresponents sinó, fem un *rollback.* Si són els corresponents realitzem un *soft delete* per marcar com a que s'ha eliminat a la base de dades però físicament encara hi existeix. Finalment, realitzem el *commit* per guardar els canvis.

11.1.4.4.4 Controller.php

Aquesta classe s'encarrega d'estendre de la classe BaseController i d'utilitzar AutrhorizesRequests, DispatchesJobs i ValidatesRequests.

[12] marcar com a que està eliminat però encara existeix a la base de dades.

11.1.4.4.5 CourseController.php

Aquesta classe s'encarrega de modificar la taula courses de la base de dades. Conté la següent funció:

- **update():** Comencem creant la transacció per la base de dades. A continuació, es realitza un bucle utilitzant el JSON que ens passen per tal d'obtenir els cursos i modificar la família dels cursos que volem. El JSON que ens entren té el següent format:

  ```
  [{"id": 0, "family": "Otros", "name": "multi-course"},
  {"id": 589, "family": "Administració",
  "name": "Técnico en Gestión Administrativa"},
  {"id": 591, "family": "Administración",
  "name": "Técnico Superior en Asistencia a la Dirección"},
  {"id": 592, "family": "Administración",
  "name": "Técnico Superior en Administración y Finanzas"}]
  ```

  Seguidament, si tot anat bé realitzem el *commit* adient per tal de guardar els canvis. Si no ha anat bé realitzem un *rollback* per desfer aquests canvis que estem produint. Finalment retornem un JSON. Un exemple de format retornat seria el següent :

  ```
  [{"id":0,"family":"Otros","name":"multi-course"},
  {"id":589,"family":"Administraci\u00f3",
  "name":"T\u00e9cnico en Gesti\u00f3n Administrativa"}
  ,{"id":591,"family":"Administraci\u00f3n",
  "name":"T\u00e9cnico Superior en Asistencia a la Direcci\u00f3n"}]
  ```

11.1.4.4.6 QueueController.php

Aquesta classe s'encarrega de crear noves cues a la taula *queues* de la base de dades. Conté la següent funció:

- **create():** Es comença creant la transacció per la base de dades. Una vegada creada la transacció omplim el *json* amb els camps *agent_id* i *student_id*. Un exemple del format que rebem seria:

  ```
  [{"id": "C100"}, {"id": "C6524"}, {"id": "C6967"},
  {"id": "C8741"}, {"id": "C8766"}, {"id": "C8767"}]
  ```

A continuació, ho guardem a la base de dades fent un *commit*, si fos el cas que quelcom anat malament realitzaríem un *rollback* el qual ens permet desfer els canvis realitzats anteriorment i llençaríem un missatge d'excepció. Finalment retornaríem un JSON. Un exemple de format retornat seria el següent:

```
[{"id":"C100"},{"id":"C6524"},{"id":"C6967"},
{"id":"C8741"},{"id":"C8766"},{"id":"C8767"},
{"id":"C9003"}]
```

11.1.4.4.7 StudentController.php

Aquesta classe s'encarrega de mostrar, per mitjà de les següents funcions, tot el contingut referent als estudiants.

- **read():** Aquesta funció s'encarrega, rebent per paràmetre un identificador d'un estudiant, de retornar un JSON. Un exemple de format sobre el JSON que retorna seria:

  ```
  [{"id":1,"name":"Miquel","last_names":"Apellido1 Apellido2",
  "personal_phone_id":605236598,"personal_mail":"miquel@example.com",
  "corporate_mail":"miquel@examplecorporate.com"}]
  ```

- **readStudents():** Aquesta funció s'encarrega de retornar un JSON amb la següent informació sobre els estudiants cridant la funció readAll(). Exemple:


  ```
  [{"id":1,"name":"Oriol","last_names":"Apellido1 Apellido2",
  "personal_phone_id":605236598,"personal_mail":"oriol@example.com",
  "corporate_mail":"oriol@examplecorporate.com"},
  {"id":2,"name":"Buba","last_names":"Apellido1 Apellido2",
  "personal_phone_id":603256987,"personal_mail":"buba@example.com",
  "corporate_mail":"buba@examplecorporate.com"}]
  ```


  A més, aquesta funció també ens permet la opció de filtrar els estudiants per semestre cridant a la funció readSemester(). De manera que podem obtenir tots els estudiants en un semestre determinat. Exemple del JSON retornat:


  ```
  [{"id":1,"name":"Rosario","last_names":"Apellido1 Apellido2",
  "personal_phone_id":605236598,"personal_mail":"rosario@example.com",
  "corporate_mail":"rosario@examplecorporate.com",
  "enrolls":[{"id":1,"course_id":1,"semester_id":1,"shop_id":1,
  "agreement":null,"fct":0,"premium":1}]}]
  ```


- **readAll():** Aquesta funció s'encarrega de retornar tots els estudiants.
- **readSemester():** Aquesta funció s'encarrega de retornar els estudiants d'un semestre en concret a la funció readStudents().
- **getCurrentAssignments():** Aquesta funció s'encarrega de tractar fent uns càlculs estadístics sobre l'assignació actual d'agents i estudiants.
- **getCurrentSpecialAssignments():** Aquesta funció s'encarrega d'obtenir els estudiants que són premium o de conveni (Ex: Sanitas, altres grups) i els hi passa la funció getCurrentAssignments() la qual els tractarà per realitzar els càlculs adients per obtenir l'assignació agent-estudiant.
- **getCurrentStandardAssignments():** Aquesta funció s'encarrega d'obtenir els estudiants que no són premium i tampoc de conveni. Els hi passa a la funció getCurrentAssignments() tenint en compte per la distribució la familia, la botiga/col·legi, si pertanyen a fct o no i el número d'estudiants totals. Seguidament, es crida a la funció getCurrentAssignments() per realitzar els càlculs adients per obtenir l'assignació agent-estudiant.
- **proposeAssignments():** Aquesta funció s'encarrega de cridar les funcions getLessBusyAgent() i updateAgentAssignmentsStats(). Crida a aquestes funcions per tal d'actualitzar estadístiques i filtrar per obtenir l'agent que estigui més lliure.
- **proposeSpecialAssignments():** Aquesta funció s'encarrega de distribuir els estudiants actuals i els que proposa l'algorisme de distribució d'assignacions agent-student, que són premium o de conveni (Ex: Sanitas, altres grups) a uns agents en específic.
- **proposeStandardAssignments():** Aquesta funció s'encarrega de distribuir els estudiants actuals i els que proposa l'algorisme, entre els agents de forma equilibrada tenint en compte: el número d'estudiants de cada agent, de si pertanyen a fct o no, de quin grup de botiga pertanyen i quina família de cicle o grau estan cursant.
- **updateAgentAssignmentsStats():** Cada cop que assignem un estudiant a un agent actualitzem les estadístiques que utilitza la funció getLessBusyAgentsByProp().
- **getLessBusyAgentsByProp():** Mira quins agents tenen menys càrrega de la propietat que se li passa per paràmetre.

- **getLessBusyAgent():** Mira quins agents tenen menys càrrega de FCT , dintre d'aquests mira quins tenen menys càrrega d'estudiant, segons la família del curs i dintre d'aquests mira quins tenen menys càrrega de grup de botigues. Finalment retorna un agent. Si n'hi ha més d'un s'agafa qualsevol utilitzant un random.

- **getProposedAssignments():** Aquesta funció s'encarrega de cridar a les funcions proposeStandardAssignments() i proposeSpecialAssignments() per tal d'obtenir els estudiants actuals + els resultants proposats.

- **getQueueFromPhone():** En funció del telèfon es retorna la cua a la qual MasVoz dirigirà la trucada.

- **normalizeWord():** S'encarrega de substituir els caràcters accentuats i altres caràcters especials, per la versió sense caràcters especials. A més, s'encarrega de convertir tots els caràcters a minúscules per tal de poder fer la cerca d'estudiants més fàcil utilitzant la funció findByName().

- **findByName():** Aquesta funció s'encarrega de realitzar una cerca aproximada utilitzant la funció levenshtein() sobre el nom i els cognoms dels estudiants. De manera que faciliti a l'agent la cerca d'un alumne si en aquell moment solament s'enrecorda per exemple del nom d'un alumne. Un exemple seria si un agent posa david com a nom d'alumne es retorna un JSON amb la següent informació:


```
[{"id":459,"name":"Eustaquio","last_names":"Apellido1 Apellido2",
"personal_phone_id":null,"personal_mail":"eustaquio@example.com",
"corporate_mail":null,"lev":1}, {"id":1107,"name":"Emmerejildo",
"last_names":"Apellido1 Apellido2 ","personal_phone_id":null,
"personal_mail":"emmerejildo@example.com", "corporate_mail":null,"lev":1}]
```


11.1.5 Tests de requeriments

En aquest apartat es mostren diferents imatges de l'aplicació amb la intenció de demostrar que els requeriments s'adapten als demanats per part del client en la primera iteració.

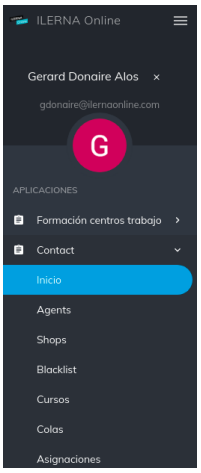

Figura 3: Menú de l'aplicació.

En aquesta imatge se'ns mostra l'aplicació amb el menú que es demanava al requeriment R.1.1. A més, també hi ha el botó *Shops* ja que ens esperàvem que a la següent iteració voldrien que sortís al menú. En conseqüència, la funcionalitat d'aquest botó no està implementada en aquesta iteració.

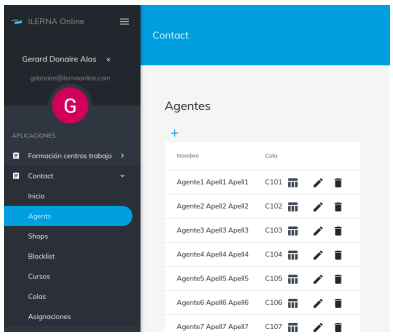

Figura 4: Informació sobre els *agents*.

Com bé veiem en aquesta imatge, quan seleccionem el botó *Agent* del menú se'ns obra una pantalla en la qual se'ns mostra els noms, cognoms dels agents i la cua a la que pertanyen. A més, tenim l'opció de crear, modificar i eliminar *agents*. Per últim, també tenim l'opció d'exportar un excel. Per tant, es compleix el requeriment R.1.2 demanat pel client.

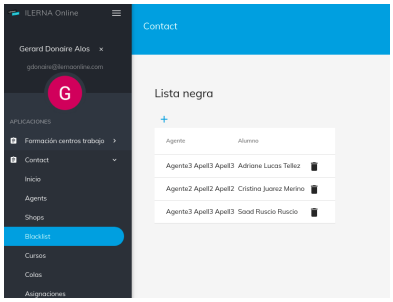

Figura 5: Informació sobre els *agents* i els alumnes que no poden estar junts.

Com bé observem en aquesta imatge, quan seleccionem el botó *Blacklist* del menú se'ns obra una pantalla en la qual se'ns mostra tant els noms, cognoms dels agents com dels alumnes que tenen problemes amb el seu agent corresponent i viceversa. A més, tenim l'opció tant d'afegir com d'eliminar una *blacklist*. Per tant, es compleix el requeriment R.1.3.

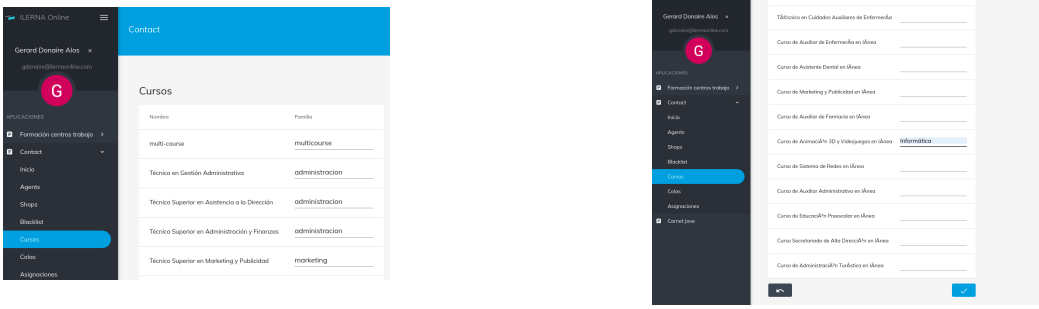

Figura 6: Pantalla d'informació sobre els cursos i botons de guardar i tirar endarrere.

En aquestes dues imatges se'ns mostra una nova pantalla la qual es manifesta un cop seleccionat el botó Cursos del menú. Se'ns presenta els noms dels cursos i els noms de la família a la qual pertany cada curs. A més, podem modificar la família del curs i disposem de dos botons. (Un per desar els canvis i l'altre per tirar endarrere). Així que podem dir que es compleix el requeriment R.1.4.

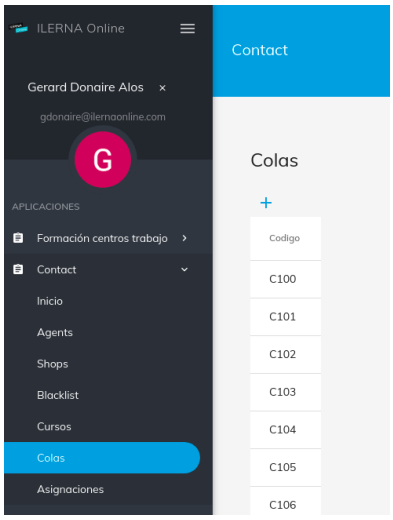

Figura 7: Pantalla d'informació sobre les cues.

En l'anterior figura se'ns manifesta una nova pantalla la qual es mostra un cop seleccionada l'opció Cues del menú. Se'ns mostra tots els noms de les cues i també l'opció de crear-ne una de nova. Així doncs, podem dir que es compleix el requeriment R.1.5.

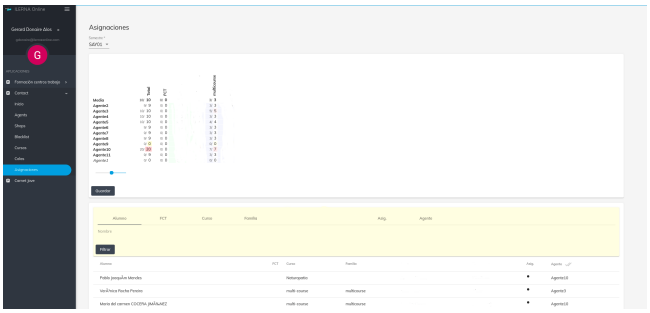

Figura 8: Pantalla d'assignacions amb els filtres demanats en l'iteració 1.

En aquesta figura se'ns presenta una nova pantalla la qual es manifesta un cop seleccionat el botó Assignacions. Se'ns mostra una taula amb els noms dels agents i amb tots els criteris que es demanava al requeriment R.1.6 tenint en compte la distribució dels alumnes que es demanava al requeriment R.5.1. Per tant, es compleixen els requeriments R.1.6 i R.5.1.

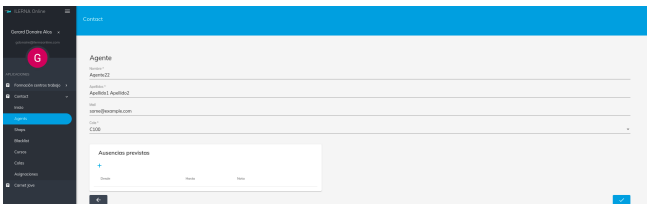

Figura 9: Pantalla de creació *agents*.

En l'anterior imatge se'ns mostra una nova pantalla la qual es manifesta un cop seleccionada l'opció crear agents amb els camps corresponents els quals són esmentats al requeriment R.2.1. A més, disposem de dos botons un per desar canvis i l'altre per tirar endarrere tal i com es demanava. En conseqüència, podem afirmar que el requeriment R.2.1 es compleix.

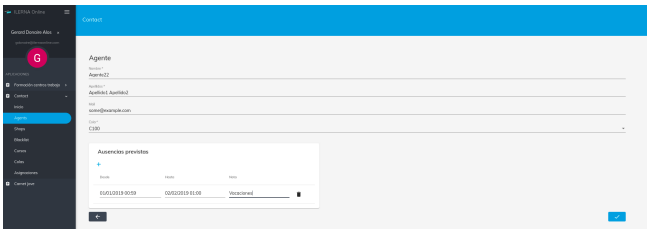

Figura 10: Pantalla de modificació d'*agents*.

En aquesta imatge se'ns mostra una nova pantalla la qual es manifesta un cop seleccionada l'opció modificar agents amb els camps corresponents els quals són esmentats al requeriment R.2.2. A més, disposem de dos botons un per desar canvis i l'altre per tirar endarrere tal i com es demanava. En conclusió, podem afirmar que el requeriment R.2.2 es compleix.

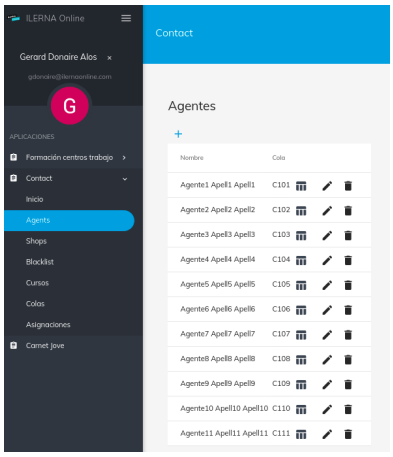

Figura 11: Pantalla d'eliminació d'*agents*.

En aquesta figura se'ns presenta l'opció, un cop seleccionada l'opció d'eliminar agents, d'eliminació d'un agent sempre que no tingui cap assignació realitzada. Per tant, es compleix el requeriment R.2.3.

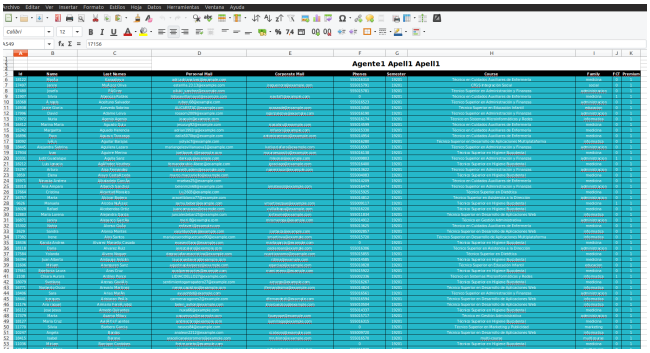

Figura 12: Excel exportat amb els camps corresponents.

En la següent figura se'ns mostra l'exportació d'un excel un cop seleccionada l'opció corresponent. En aquest excel se'ns mostra els camps corresponents els quals són esmentats al requeriment R.2.4. A més, cada fulla representa el nom del semestre i estan ordenades de manera descendent. Així doncs, podem afirmar que el requeriment R.2.4 es compleix.

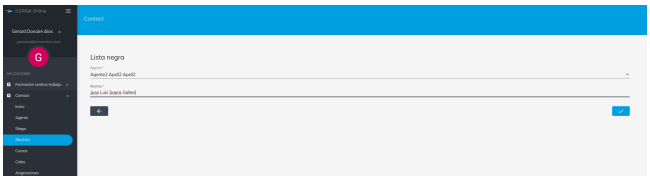

Figura 13: Pantalla de creació de *Blacklist*.

En la següent imatge se'ns manifesta una nova pantalla la qual es mostra un cop seleccionada l'opció de crear una *blacklist* amb els camps corresponents tal i com es demanava al requeriment R.3.1. A més, disposem de dos botons un per desar canvis i l'altre per tirar endarrere tal i com es demanava. En conseqüència, podem dir que es compleix el requeriment R.3.1.

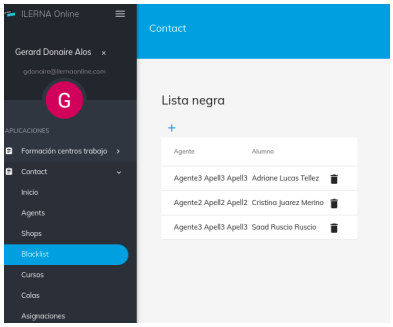

Figura 14: Opció d'eliminació d'una *Blacklist*.

En aquesta imatge se'ns presenta l'opció d'eliminació d'una *blacklist* tal i com es demanava en el requeriment R.3.2.

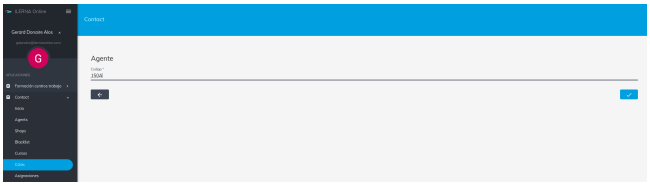

Figura 15: Pantalla de creació de Cues.

En la següent imatge se'ns mostra una nova pantalla la qual es manifesta un cop seleccionada l'opció crear cua amb els camps corresponents els quals són esmentats al requeriment R.4.1. A més, disposem de dos botons un per desar canvis i l'altre per tirar endarrere tal i com es demanava. En conclusió, podem afirmar que el requeriment R.4.1 es compleix.

Pel que fa als requeriments no funcionals, es compleixen els requeriments d'eficiència ja que el temps que es demana es el que tarda. Pel que fa al requeriment de portabilitat fins al moment sols s'ha demostrat que és suportat per *Google Chrome*. Pel que fa al requeriment no funcional de desenvolupament hem utilitzat les eines demanades.

11.2 Iteració 2

11.2.1 Definició dels Requeriments

Una vegada realitzada la primera iteració, van sorgir nous requeriments i una millora dels anteriors.

11.2.1.1 Requeriments Funcionals

Els requeriments nous que ens ha demanat el client són:

- **R.6. - Requeriments relacionats amb el seleccionament d'opcions de la nova pantalla de shops.**
  - **R.6.1 -** El sistema haurà de permetre la modificació del nom del grup existent en una nova pantalla, un cop seleccionada l'opció de modificar grup, al qual pertany una determinada botiga i si és *partner* o no. La nova pantalla ha de constar dels següents camps: nom de la botiga, nom del grup al qual pertany la botiga, una opció seleccionable de si és *partner* o no. A més, aquesta nova pantalla ha de tenir dos botons: un per desar els canvis i l'altre per tirar endarrere per si es donés el cas de que l'usuari s'equivoca.
  - **R.6.2 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó *Shops* del menú, tant el nom de les botigues com el grup a la qual pertanyen. A més, també s'haurà de mostrar de si és *partner* o no. Tot seguit, en la nova pantalla, hi ha d'haver alguna opció per poder modificar el grup al qual pertany una botiga i si actualment és *partner* o no.
- **R.7. - Requeriments funcionals bàsics.**
  - **R.7.1 -** El sistema haurà de mostrar un menú amb els següents camps: *Agents, Shops, Blacklist*, Cursos, Cues, Assignacions. (Com podem observar s'ha afegit el camp *Shops*.)
  - **R.7.2 -** El sistema haurà de mostrar, en una nova pantalla un cop s'hagi seleccionat el botó Assignacions del menú, una taula amb el nom dels agents i amb els següents criteris: el número total d'estudiants que té cada agent amb la mitjana d'estudiants que haurien de tenir com a màxim. El número d'estudiants que té cada agent per FCT, família del cicle (Sanitat, Informàtica, etc), grup de botiga i la repartició d'alumnes per agent ha de ser proporcional pel tipus de criteri. A més, es disposarà d'un llistat sense filtrar inicialment on es veurà tots els alumnes registrats a la seva cartera. Disposarà d'uns filtres concrets per poder facilitar la cerca. (Alumne, FCT, Curs, Família, *Shop, Shop Groups*, Assignacions i Agent).

- **R.8. - Requeriments relacionats amb la distribució dels alumnes entre els agents.**
  - **R.8.1 -** El sistema haurà de permetre l'assignació d'alumnes als agents tenint en compte els següents criteris:
    * Hi hauran dos tipus d'agents: agents especials i agents. Els agents especials s'encarregaran dels alumnes que són *premium* i/o pertanyen de centres col·laboradors. Els agents s'encarregaran de la resta d'alumnes que no pertanyen ni a premium ni a centres col·laboradors.
    * Per distribuir els alumnes entre els agents normals s'ha de considerar el següents criteris amb un ordre de priotitat:
      · El criteri amb més priotitat: el nombre d'alumnes que estan cursant FCT que tenen els agents. L'agent amb el menor nombre d'alumnes serà al que se li assignarà l'alumne. En cas d'empat entre agents es passa al segon criteri amb més prioritat.
      · El segon criteri amb més prioritat: el nombre total d'alumnes que tenen els agents. L'agent amb el menor nombre total d'alumnes serà al que se li assignarà l'alumne. En cas d'empat entre agents es passa al tercer criteri amb més prioritat.
      · El tercer criteri amb més prioritat: el nombre d'alumnes, que tenen els agents, del grup de botiga al qual pertany l'alumne. L'agent amb el menor nombre d'alumnes del grup de botiga que pertany l'alumne serà al que se li assignarà aquest alumne. En cas d'empat entre agents es passa a l'últim criteri.
      · Últim criteri: el nombre d'alumnes, que tenen els agents, de la família del cicle al qual pertany l'alumne. L'agent amb el menor nombre d'alumnes de la família del cicle al qual pertany l'alumne serà al que se li assignarà aquest alumne. En cas d'empat entre agents s'haurà d'escollir aleatòriament entre els que han empatat.

11.2.2 Planificació

En aquesta part de la segona iteració, és on esmentarem la planificació de la segona part del projecte per tal de finalitzar la estructura del projecte. La planificació és la part més important ja que les següents fases d'aquesta iteració final partiran sobre aquesta.

Pel que fa a la planificació del projecte vam decidir que la part de backend la continuaríem realitzant en Marc Olivé i jo amb el llenguatge de programació PHP (usant com a PHP IDE PhpStorm per aquest llenguatge), utilitzant com a *framework* Laravel Eloquent i la base de dades de MySQL. En canvi, la part de frontend la farà l'Ismael López Pérez.

Hem continuat utilitzant aquestes eines ja que són les eines en que es treballen a la pròpia empresa.

11.2.3 Disseny

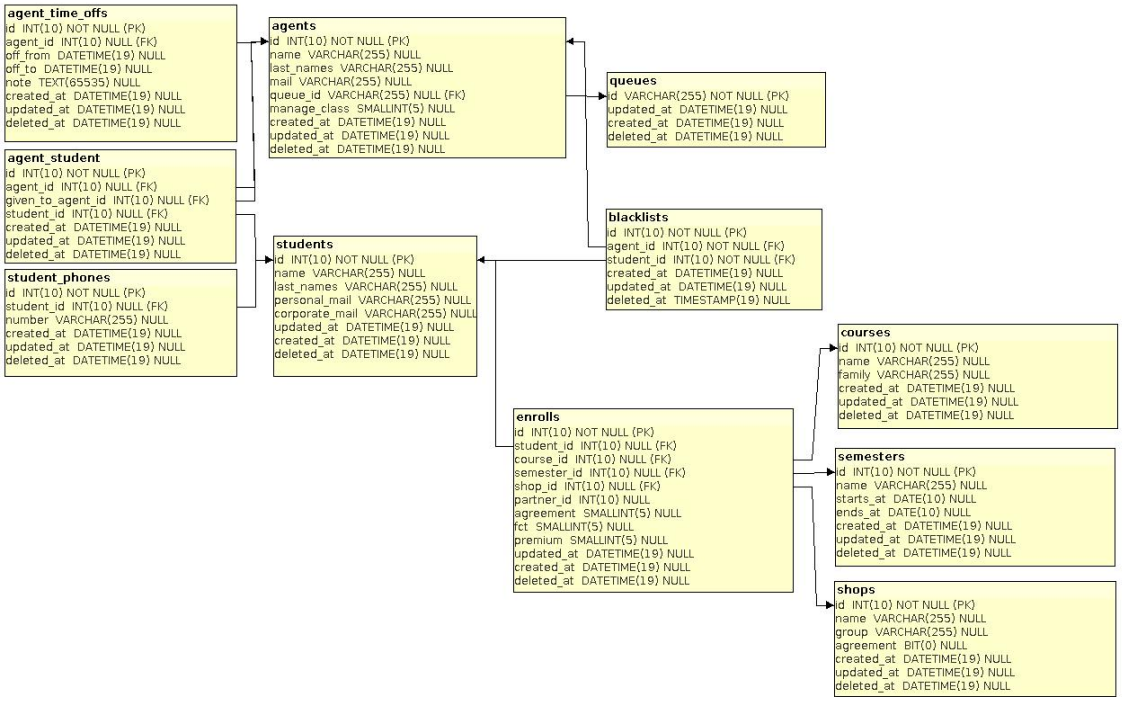

Figura 16: Tercera versió del Diagrama UML sobre la base de dades de l'aplicació.

Com bé podem veure en la figura 1 vam començar realitzant les taules que creiem que eren necessàries però més tard, com bé podem veure en la figura 2, vam treure les taules *agreements*, *partners* i *shops* ja que vam veure que era innecessari tenir tres taules quan les podíem agrupar-ho tot en una. Aquesta nova figura 16, reflecteix que al tenir els nous requeriments corresponents a *shops* el que vam fer es crear la taula *shops* amb els nous camps *group* i *agreement* (on el *group* representa el nom del grup de la botiga (*partner*) i l'*agreement* l'utilitzem a la taula *enrolls* per tal de saber si pertany a un conveni o no. També l'utilitzem a la taula *shops* per poder mostrar-ho a la vista de l'aplicació.

Finalment, vam obtenir l'UML final sobre la Base de Dades on el podem veure a l'annex en la figura 23. En aquesta figura solament hem afegit una taula *users* que ens serveix per restringir qui pot accedir a la nostra aplicació per qüestions de seguretat. A més, un cop final·litzada la implementació de les dues iteracions vam obtenir el següent diagrama UML sobre els models i els controladors de la nostra aplicació. Ho podem veure en la següent figura 22 de l'annex.

11.2.4 Implementació

En aquesta part de la segona iteració, un cop acabada la part de disseny, esmentarem la implementació [13] que hem dut a terme. És a dir, el desenvolupament dels Models, dels Controladors en classes. Esmentarem tant les funcions que conté cada model i controlador com les seves rutes corresponents. També parlarem sobre el desenvolupament de les classes realitzades per a la migració a la Base de Dades i finalment mostrarem els testos adients que han servit per una mera comprovació sobre la implementació. Començarem explicant les rutes, seguidament els models, després els controladors, més tard les classes per a la migració a la Base de Dades i finalment els testos sobre la implementació.

[13] traduir el disseny en una forma legible per la màquina.

11.2.4.1 Rutes Shops

```php
<?php
    Route::middleware('auth:api')
    -> get('/shops', function()
    {
        return App\Shop::all();
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus GET i retornar els camps:*id, name, group* fent ús de la funció all() del *framework* Laravel i utilitzant així el model *Shop*.

```php
<?php
    Route::middleware('auth:api')
    -> put('/shops', function(Request $request)
    {
        return App\Http\Controllers\ShopController::
        update($request->all());
    });
?>
```

Aquesta ruta s'encarrega de rebre una petició HTTP del client de tipus PUT i retornar els camps: *id, name, group* incloent els modificats fent ús de la funció **update()** del controlador ShopController.

11.2.4.2 Models

Totes les següents classes estenen de la classe Model exceptuant les classes Template.php i User.php.

11.2.4.2.1 Shop.php

Aquesta classe igual que les anteriors també té les dues variables **$hidden** i **$fillable** per tal de filtrar el contingut desitjat.

11.2.4.3 Controllers

Totes les següents classes estenen de la classe Controller.

11.2.4.3.1 ShopController.php

Aquesta classe s'encarrega de modificar el grup al qual pertany cada botiga. Conté la següent funció:

- **update():** es comença creant la transacció per la base de dades. A continuació, recollim un JSON que ens envien. Un exemple de JSON que rebem seria:

  ```
  [{"id": 0, "name": "multi-shop", "group": "Otros", "agreement": 0},
  {"id": 1, "name": "ILERNA Online", "group": "Ilerna", "agreement": 0}]
  ```

  Seguidament, ho guardem a la base de dades (*id, group, agreement* (ja que el *name* simplement el retornem no el guardem a la base de dades)) amb un save() i realitzem el *commit* per tal d'efectuar els canvis. Si fos el cas que tinguéssim algun problema a l'hora de guardar-ho a la base de dades realitzaríem un *rollback* per desfer els canvis i llençaríem un missatge d'error. Finalment, retornem un JSON. Un exemple de format retornat seria:

  ```
  [{"id":0,"name":"multi-shop","group":"Otros","agreement":0},
  {"id":1,"name":"ILERNA Online","group":"Ilerna","agreement":0}]
  ```

11.2.4.4 Classes per a les migracions de BD.

Les migracions són com el control de versions per a la nostra base de dades, el que ens permet modificar fàcilment i compartir l'esquema de la base de dades de l'aplicació. Laravel ens proporciona un suport de base de dades per tal de crear i manipular taules a través de tots els sistemes de base de dades compatibles amb Laravel. Les migracions es combinen amb el constructor d'esquemes de Laravel per construir fàcilment l'esquema de la base de dades de la nostra aplicació.

Una classe per a la migració de BD conté dos mètodes: up() i down(). El mètode up() s'utilitza per afegir noves taules, columnes o índexs a la nostra base de dades, mentre que el mètode down() hauria de revertir les operacions realitzades pel mètode up().

Dins d'aquests dos mètodes, podem utilitzar el constructor d'esques de Laravel per crear i modificar expressament taules.

En el cas d'aquest projecte hem creat les migracions de les següents classes:

- **Agents.**
- **Queues.**
- **AgentTimeOffs.**
- **Students.**
- **StudentPhones.**
- **Courses.**
- **AgentStudent.**
- **Blacklists.**
- **Semesters.**
- **Shops.**
- **Enrolls.**

A continuació mostraré un exemple de classe que he implementat per a les migracions de BD.

```php
<?php
    class Agents extends Migration
    {
        /**
        * Run the migrations.
        *
        * @return void
        */
        public function up()
        {
            Schema::create('agents', function (Blueprint $table)
            {
                $table->increments('id');
                $table->string('name', 255);
                $table->string('last_names', 255);
                $table->string('mail', 255)->nullable();
                $table->string('queue_id', 255)->charset('ascii');
                $table->smallInteger('manage_class')->default(0);

                $table->dateTime('created_at');
                $table->dateTime('updated_at');
                $table->dateTime('deleted_at')->nullable();

                $table->foreign('queue_id')->references('id')->on('queues');

                $table->engine = 'InnoDB';
                $table->charset = 'utf8';
                $table->collation = 'utf8_unicode_ci';
            });
        }

        /**
        * Reverse the migrations.
        *
        * @return void
        */
        public function down()
        {
            Schema::dropIfExists('agents');
        }
    }
?>
```

Com bé veiem en aquest exemple, a la funció up() estem creant una taula anomenada *agents* amb els seus camps corresponents (*id, name, last_names, queue_id, manage_class, created_at, updated_at, delated_at*). El camp id serà incremental és a dir a mesura que es vagi inserint un agent, el camp id anirà sumant 1. El *name, last_names, mail* corresponen al nom, cognoms, correu electrònic corresponent a l'agent. El camp *queue_id* representa l'identificador de cua al qual està associat aquest agent per tal d'introduir-lo a una cua. Els camps *created_at i updated_at* per defecte agafarà la data de creació i la data de modificació respectivament. (el temps en el qual es crea i es modifica la taula.) Per últim el camp *deleted_at* serà per defecte *null*.

La funció down() simplement elimina els canvis produïts en la funció down(). Dit amb altres paraules, elimina la taula que havíem creat si existeix.

11.2.5 Tests de requeriments

En aquest apartat es mostren diferents imatges de l'aplicació amb la intenció de demostrar que els requeriments s'adapten als demanats per part del client en la segona iteració.

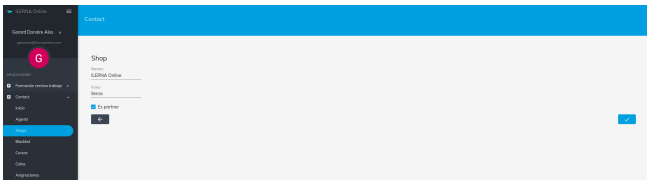

Figura 17: Pantalla de modifiació de *Shops*.

En aquesta imatge se'ns mostra l'opció per modificar el nom del grup existent en una nova pantalla un cop seleccionada l'opció de modificar el grup al qual pertany una determinada *shop* i també es mostra si és *partner* o no. A més, la nova pantalla consta dels camps corresponent tal i com es demanava al requeriment R.6.1 i amb els dos botons adients. Així que podem afirmar que es compleix el requeriment R.6.1.

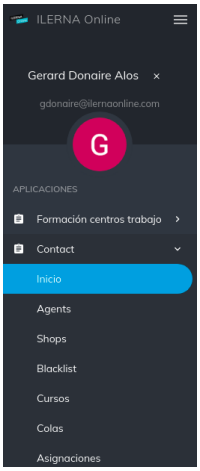

Figura 18: Menú amb l'opció Shops i funcionalitat implementada.

En la següent figura se'ns mostra el menú modificat amb la nova opció *Shops*. En conseqüència, podem afirmar que el requeriment R.7.1 es compleix.

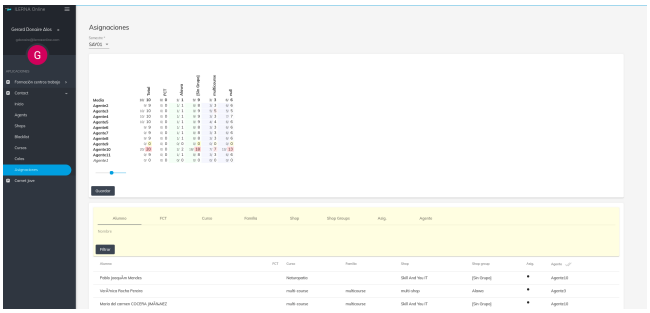

Figura 19: Pantalla d'assignacions corresponent a l'iteració 2.

En la següent imatge se'ns presenta una nova pantalla la qual es mostra un cop seleccionada l'opció d'Assignacions amb els camps corresponents tal i com es demanava al requeriment R.7.2 tenint en compte el nou criteri de distribució d'alumnes R.8.1 . A més, disposem del filtrat que es demanava. Així doncs, podem dir que es compleix el requeriment R.7.2.

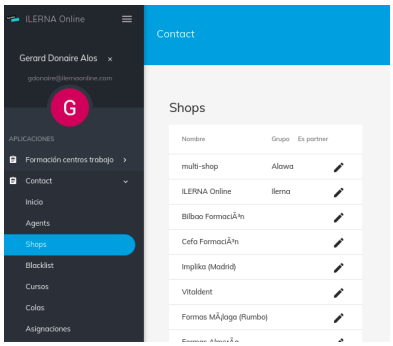

Figura 20: Pantalla corresponent a l'opció *Shops*.

En aquesta figura se'ns manifesta una nova pantalla, amb els camps corresponents, la qual es mostra un cop seleccionat el botó *Shops*. A més, es mostra si aquella *Shop* és *partner* i es mostra una opció per modificar el grup al qual pertany una botiga. Per tant, podem afirmar que es compleix el requeriment R.6.2.

12 Conclusions

La realització d'aquest treball m'ha servit per guanyar més experiència i poder créixer com a professional en el món laboral d'una empresa. A part d'aprendre sobre com es treballa avui en dia en una empresa, també he agafat experiència com a programador aprenent nous llenguatges com *Php* i un nou *framework* anomenat Laravel. A més, també he après com funciona tant la part de *backend* com la de *frontend* a l'hora de realitzar aplicacions web.

Podríem dir que m'ha servit per familiaritzar-me més amb l'enginyeria de *software* veient quines metodologies utilitzen i com es planifiquen els projectes. Trobo encertada la metodologia *Agile* que utilitzen a l'empresa Ilerna Online ja que per realitzar projectes de *software* considero que és la millor per la facilitat que dóna tant al programador com al client que fa la petició del projecte en termes d'organització. A més de la facilitat que ens dóna aquesta metodologia, l'organització que ens proporciona crec que és el que més m'agrada d'aquesta metodologia ja que en tot moment es sap que està realitzant cadascun dels programadors, quina feina tenen, com solucionen els problemes que tenen, etc. Crec que per qualsevol departament és molt important el fet de treballar en equip i seguir metodologies com l'*Agile*. I el motiu pel qual penso així és pel fet que una metodologia com l'*Agile* ens proporciona una organització estructurada sobre les tasques que ha de realitzar cada empleat i el fet que cadascun d'aquests empleats hagi de parlar amb els del seu equip fa que augmenti la confiança amb els altres professionals de l'equip. En conseqüència, es forma una relació satisfactòria per a l'empresa ja que si un empleat està satisfet amb l'equip acostuma a ser més productiu.

En general he pogut observar com s'estructura un projecte en etapes i aquestes etapes en que consisteixen cadascuna. També, he assolit el coneixement de treballar en equip realitzant tant *sprints* com *sprints plannings*, *sprint reviews*, *sprints retrospective* i sobretot *Daily Scrums*.

Referent al llenguatge de programació *php* usat a l'empresa Ilerna Online considero que no és el meu preferit i crec que es podria utilitzar altres llenguatges com *python*. En el meu cas m'hagués agradat *python* per la experiència que tinc programant amb aquest llenguatge. Preferiria *Pyhton* abans que *PHP* ja que *Python* té una sintaxi més simple i clara la qual cosa també ajuda a que sigui més fàcil de llegir que *PHP*. Una altra característica molt lligada amb el llenguatge *PHP* seria el *framework Laravel* amb l'*ORM Eloquent*. Pel que fa a aquesta característica trobo bastant encertada la decisió d'ús tant del *framework* com l'*ORM* pel llenguatge *PHP* ja que és un dels *frameworks* més utilitzats per aquest llenguatge per la seva facilitat d'ús. A més, avui en dia aquest *framework* juntament amb el seu llenguatge és un dels més utilitzats en projectes web.

Referent a la base de dades trobo encertada la decisió d'utilitzar *MySQL* ja que per realitzar aquest projecte es necessita una base de dades mitjana i no necessitem realitzar *queries* complexes i llargues. A més, *MySQL*, és un sistema que en general és més simple i busca atreure a desenvolupadors que volen treballar amb una mica més de comoditat, amb *queries* simples i bases de dades petites o mitjanes.

Per concloure, aquest projecte m'ha servit per interioritzar i aprendre d'una manera adequada com es realitza un projecte de *software* per a un futur amb una millor noció per a l'hora de realitzar un nou projecte com aquest, els passos que s'han de dur a terme i la forma correcta per realitzar-lo. D'aquesta manera tinc una idea més formada del treball al qual es troba un veritable professional en una empresa.

13 Treball Futur

Fins al moment, com a versió final del projecte, tenim fet que un agent pugui estar de baixa en un determinat moment per causes personals. També tenim que cada agent ja disposa d'una cua i en aquesta cua hi pot haver múltiples agents. A més, la trucada de cada alumne ja es dirigeix cap a l'agent del qual correspon aquell alumne. Tanmateix controlem el fet de si s'ha de dur a terme una reassignació entre alumne-agent.

Per últim hem implementat que cada agent es pugui descarregar el seu excel pertinent. Per tant, tot el que es volia inicialment d'aquesta aplicació podríem dir que ja està realitzat. Però ara bé, com en totes les aplicacions, segurament que en un futur s'haurà de fer una ampliació sobre aquest projecte afegint per exemple a l'excel nous camps com ara les assignatures que està realitzant cada alumne. Pot ésser que es vulgui canviar per exemple la part de visualització de l'aplicació (*frontend*) en un futur. Fins i tot, es podria millorar l'aplicació per exemple optimitzant el codi per a que no es tardi tant en descarregar l'excel entre altres coses (qüestions d'eficiència).

El *software* indubtablement patirà canvis, i s'haurà de fer algunes modificacions a la seva funcionalitat. És de suma importància que el *software* de qualitat pugui adaptar-se amb la finalitat d'acoblar-se als canvis del seu entorn extern.

14 Referències bibliogràfiques

Seguidament es mostrarà tota aquella font d'informació, utilitzada fins ara, per tal de dur a terme el projecte:

Bibliografia

14.1 Consultes sobre PHP

[1] *¿Qué és PHP?*, *Data de la consulta: 9:05 el 12 de febrer de 2019*. Disponible a:
http://php.net/manual/es/intro-whatis.php
https://conceptodefinicion.de/php/
http://www.alegsa.com.ar/Dic/php.php

[2] *Manual PHP*, *Data de la consulta: 09:52 el 12 de febrer de 2019*. Disponible a:
http://php.net/manual/en/index.php

[3] *Taula de continguts*, *Data de la consulta: 09:52 el 12 de febrer de 2019*. Disponible a:
http://php.net/manual/es/tutorial.php

[4] *How can I convert string to date type in PHP?*, *Data de la consulta: 10:25 el 15 de febrer de 2019*. Disponible a: https://www.quora.com/How-can-I-convert-string-to-date-type-in-PHP

14.2 Consultes sobre SQuirrel SQL Client, MySQL i SQL

[5] *What is? What provides?* , *Data de la consulta: 10:42 el 12 de març de 2019*. Disponible a:
http://squirrel-sql.sourceforge.net/
https://www.sqlbot.co/blog/squirrel-sql-review-and-tutorial

[6] *Manual SQuirrel SQL Client*, *Data de la consulta: 13:42 el 01 de març de 2019*. Disponible a:
http://squirrel-sql.sourceforge.net/user-manual/quick_start.html

[7] *¿Qué és MySQL?*, *Data de la consulta: 09:23 el 03 de març de 2019*. Disponible a:
https://www.esepestudio.com/noticias/que-es-mysql
https://ca.wikipedia.org/wiki/MySQL

[8] *¿Qué és SQL?. Wikipedia*, *Data de la consulta: 11:17 el 06 de març de 2019*. Disponible a:
https://ca.wikipedia.org/wiki/Structured_Query_Language

14.3 Consultes sobre PhpSpreadSheet

[9] *Fatal error while using Writer*, *Data de la consulta: 10:21 el 18 de març de 2019*. Disponible a: https://github.com/PHPOffice/PhpSpreadsheet/issues/202

[10] *Recipes*, *Data de la consulta: 11:13 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/topics/recipes/

[11] *Function List By Name*, *Data de la consulta: 11:16 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/references/function-list-by-name/

[12] *Accessing cells*, *Data de la consulta: 11:16 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/topics/accessing-cells/

[13] *How to get all sheet names without trimming?*, *Data de la consulta: 13:16 el 19 de març de 2019*. Disponible a: https://stackoverflow.com/questions/52375966/how-to-get-all-sheet-names-without-trimming

[14] *PHP Spreadsheet Control*, *Data de la consulta: 11:16 el 16 de març de 2019*. Disponible a: https://www.syncfusion.com/jquery/php-ui-controls/spreadsheet

[15] *Setting a cell value by coordinate*, *Data de la consulta: 12:42 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/topics/accessing-cells/#excel-datatypes

[16] *Setting a cell value by column and row*, *Data de la consulta: 13:01 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/topics/accessing-cells/#setting-a-cell-value-by-column-and-row

[17] *Looping through cells*, *Data de la consulta: 12:01 el 19 de març de 2019*. Disponible a: https://phpspreadsheet.readthedocs.io/en/latest/topics/accessing-cells/#looping-through-cells

[18] *Sheets iterating by sheet name in alphabetical order*, *Data de la consulta: 12:05 el 19 de març de 2019*. Disponible a: https://github.com/box/spout/issues/207

[19] *Create Worksheets*, *Data de la consulta: 12:07 el 19 de març de 2019*. Disponible a: https://gist.github.com/ahgood/562090d22b6eed262ae43e878c292968

14.4 Consultes sobre Laravel Eloquent

[20] *Qué es un ORM, qué es Active Record y cómo es Eloquent en Laravel*, *Data de la consulta: 13:05 el 18 de febrer de 2019*. Disponible a:
https://desarrolloweb.com/articulos/laravel-eloquent.html

[21] *Eloquent: Relationships*, *Data de la consulta: 12:25 el 21 de febrer de 2019*. Disponible a:
https://laravel.com/docs/5.8/eloquent-relationships

[22] *Database: Getting Started*, *Data de la consulta: 09:16 el 22 de febrer de 2019*. Disponible a:
https://laravel.com/docs/5.8/database

[23] *Eloquent: API Resources*, *Data de la consulta: 10:31 el 22 de febrer de 2019*. Disponible a:
https://laravel.com/docs/5.8/eloquent-resources

[24] *BelongsToMany::withTimestamps()* , *Data de la consulta: 11:26 el 22 de febrer de 2019*. Disponible a: https://github.com/laravel/framework/issues/8662

[25] *Laravel Eloquent with 'with' and 'wherehas'*, *Data de la consulta: 10:12 el 8 de març de 2019*. Disponible a:
https://stackoverflow.com/questions/51587709/laravel-eloquent-with-with-and-wherehas
https://zaengle.com/blog/using-wherehas-in-laravel-polymorphic-relations
https://stackoverflow.com/questions/25179155/laravel-eloquent-orm-wherehas-through-table
https://github.com/laravel/framework/issues/18415

[26] *orderBy on hasMany relationship!*, *Data de la consulta: 09:17 el 23 de març de 2019*. Disponible a: https://laravel.io/forum/09-14-2014-orderby-on-hasmany-relationship

[27] *Alias para consulta en dos tablas con laravel*, *Data de la consulta: 12:01 el 25 de març de 2019*. Disponible a:
https://es.stackoverflow.com/questions/93152/alias-para-consulta-en-dos-tablas-con-laravel

[28] *Retrieve Eloquent model using multiple table query*, *Data de la consulta: 11:50 el 27 de març de 2019*. Disponible a:
https://laravel.io/forum/11-25-2014-retrieve-eloquent-model-using-multiple-table-query

[29] *Qué es Query Builder, junto con una guía de uso en el framework PHP Laravel 5.1.*, *Data de la consulta: 12:43 el 22 de març de 2019*. Disponible a:
https://desarrolloweb.com/articulos/query-builder-laravel5.html

[30] *Consultar múltiples tablas relacionadas en Laravel (Eloquent).*, *Data de la consulta: 09:12 el 24 de març de 2019*. Disponible a:
https://rimorsoft.com/consultar-multiples-tablas-relacionadas-en-laravel-eloquent

[31] *Raw Queries in Laravel*, *Data de la consulta: 12:10 el 03 de març de 2019*. Disponible a:
https://fideloper.com/laravel-raw-queries

[32] *Raw Queries in Laravel*, *Data de la consulta: 12:10 el 03 de març de 2019*. Disponible a:
https://fideloper.com/laravel-raw-queries

[33] *How to whereHas when use belongsToMany relation?*, *Data de la consulta: 11:05 el 11 de març de 2019*. Disponible a:
https://laracasts.com/discuss/channels/eloquent/how-to-wherehas-when-use-belongstomany-relation

[34] *Sorting Parent Models By a Child Relationship*, *Data de la consulta: 10:21 el 15 de març de 2019.* Disponible a:
https://zaengle.com/blog/sorting-parent-models-by-a-child-relationship

[35] *Order by on relationship field*, *Data de la consulta: 13:10 el 27 de març de 2019.* Disponible a:
https://laracasts.com/discuss/channels/eloquent/order-by-on-relationship

[36] *¿Qué frameworks usan las empresas?*, *Data de la consulta: 09:10 el 28 de març de 2019.* Disponible a: https://www.hrcs.es/noticias/2017/09/13/php-frameworks/

14.5 Consultes sobre Servidor Apache

[37] *¿Qué es Apache? Descripción completa del servidor web Apache*, *Data de la consulta: 10:12 el 06 de març de 2019.* Disponible a: https://www.hostinger.es/tutoriales/que-es-apache/

[38] *¿Qué hace un Servidor Web como Apache?. Configuración*, *Data de la consulta: 12:15 el 03 de març de 2019.* Disponible a:
https://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/.

14.6 Consultes sobre HTML

[39] *Definición de HTML*, *Data de la consulta: 12:12 el 01 de març de 2019.* Disponible a:
https://definicion.de/html/.
https://devcode.la/blog/que-es-html/.

14.7 Consultes sobre PhpStorm

[40] *FEATURES*, *Data de la consulta: 12:15 el 03 de març de 2019.* Disponible a
https://www.jetbrains.com/phpstorm/features/.

[41] *¿Qué és PhpStorm?. Wikipedia i JetBrains*, *Data de la consulta: 09:23 el 29 de febrer de 2019.* Disponible a:
https://en.wikipedia.org/wiki/PhpStorm.
https://comprasoft.com/jetbrains/phpstorm.
https://ca.wikipedia.org/wiki/Entorn_integrat_de_desenvolupament

14.8 Consultes sobre Metodologia Agile

[42] *Metodologia Cascada vs Agile* , *Data de la consulta: 09:13 el 21 de març de 2019.* Disponible a:
https://comunidad.iebschool.com/ladyintech/2017/05/18/metodologias-en-cascada-vs-agile/
https://medium.com/forecast-en-espa%C3%B1ol/agile-vs-cascada-parte-1-de-5-qu%C3%A9-es-la-metodolog%C3%ADa-en-cascada-cc0ad7ea9875

[43] *Qué son las metodologías de trabajo ágiles*, *Data de la consulta: 11:11 el 29 de març de 2019.* Disponible a: https://www.ennaranja.com/agile/el-auge-de-las-metodologias-agiles/

[44] *Principios en los que se basa la metodología Agile*, *Data de la consulta: 11:15 el 29 de març de 2019.* Disponible a:
https://www.obs-edu.com/es/blog-project-management/agile-project-management/principios-en-los-que-se-basa-la-metodologia-agile

14.9 Consultes sobre Programació per Capes

[45] *Qué es la Programación por capas, Capas y niveles, Data de la consulta: 12:35 el 23 de març de 2019*. Disponible a: https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

15 Annex

15.1 Definicions

- **IT:** és un terme que es refereix a maquinari, programari, telecomunicacions, xarxes i persones involucrades per crear, emmagatzemar, intercanviar i utilitzar la informació.
- **PHP:** és un llenguatge interpretat d'alt nivell que pot ser afegit en el llenguatge HTML. Aquest llenguatge és de codi obert, i especialment adequat per al desenvolupament web. Quan un client (qualsevol persona en la web) visita la pàgina web que conté aquest codi, el servidor l'executa i el client solament rep el resultat. La seva execució és per tant al servidor, a diferència d'altres llenguatges de programació que s'executen al navegador.
- **HTML**: és un llenguatge de marcat dissenyat per estructurar textos i relacionar-los en forma d'hipertext. S'encarrega de desenvolupar una descripció sobre els continguts que apareixen com a text i sobre la seva estructura. És un llenguatge molt simple i general que serveix per definir altres llenguatges que tenen que veure amb el format dels documents. El text en ell es crea a partir d'etiquetes, també anomenades tags, que permeten interconnectar diferents conceptes i formats. Gràcies a Internet i als navegadors web, s'ha convertit en un dels formats més populars que existeixen per a la construcció de documents per a la web.
- **PhpStorm:** és un PHP IDE intel·ligent i professional que proporciona als desenvolupadors una combinació de potents eines intel·ligents, hàbits útils i millores pràctiques per al desenvolupament de PHP, tots orientats per beneficiar a la productivitat del desenvolupador.
- **PHP IDE:** és una eina informàtica per al desenvolupament de programari de manera còmoda i ràpida. Així doncs és un entorn de desenvolupament que agrupa diferents funcions en un sol programa, habitualment: editor de codi, compilador, depurador i un programa de disseny d'interfície gràfica. El principal avantatge és que facilita la tasca del programador mentre que l'inconvenient més important és que pot provocar mals hàbits a l'hora de programar o provocar errors que a priori començant de zero no es produirien.
- **MySQL:** és un sistema d'administració de bases de dades per a bases de dades relacionals. A més, és multifil i multiusuari, que utilitza el llenguatge SQL. Així que és una aplicació que permet gestionar fitxers cridats de bases de dades.
- **SQL:** és un llenguatge estàndard de comunicació amb bases de dades relacionals. És a dir, un llenguatge normalitzat que permet treballar amb la majoria de bases de dades relacionals. La principal característica d'aquest llenguatge és la seva simplicitat, ja que amb pocs coneixements es poden fer consultes bàsiques sobre una base de dades.
- **SQuirrel SQL Client:** és una eina d'administració de bases de dades. Permet als usuaris explorar i interactuat amb bases de dades.
- **Laravel Eloquent:** és un *framework* de codi obert per desenvolupar aplicacions i serveis web amb PHP. La seva filosofia és desenvolupar codi PHP de forma elegant i simple. Està basat en el model MVC.
- **PhpSpreadSheet:** és una biblioteca escrita en PHP pur i proporciona un conjunt de classes que ens permeten llegir i escriure en diferents formats de fitxer de fulls de càlcul, com Excel i LibreOffice Calc.
- **MVC:** és un patró de disseny utilitzat per a la implementació d'interfícies d'usuari. Aquest patró de desenvolupament de programari divideix l'aplicació en tres parts interconnectades: el model de dades, la interfície usuari i la lògica de control. El patró MVC es veu freqüentment en aplicacions web, on es pot visualitzar una pàgina HTML i el codi que proveeix de dades dinàmiques a la pàgina, el controlador és el sistema de gestió de bases de dades i el model és el model de dades.

- **FCT:** és el mòdul que els alumnes parteixen al final dels seus estudis de Formació Professional i, que es porta a terme, gràcies a la col·laboració de les empreses, ja que es desenvolupa en la seva totalitat en les pròpies empreses.
- **Contact:** persones que s'encarreguen de l'atenció dels alumnes.
- **JSON:** és un estàndard obert basat en text dissenyat per a intercanvi de dades llegible per humans. Deriva del llenguatge script JavaScript, per a representar estructures de dades simples i llistes associatives, anomenades objectes. El format JSON s'utilitza habitualment per serialitzar i transmetre dades estructurades en una connexió de xarxa. S'utilitza principalment per intercanviar dades entre un servidor i una aplicació web, sent una alternativa a l'XML. S'utilitza freqüentment en aplicacions Ajax.
- **JavaScript:** és un llenguatge de programació que ens permet crear contingut nou i dinàmic, controlar fitxers de multimèdia, crear imatges animades i moltes altres coses més.
- **Objecte:** en informàtica, en la programació orientada a objectes, cada element que té un significat per si mateix, que té mètodes que determinen el seu comportament i propietats que el defineixen. Els objectes són instàncies d'una classe.
- **Classe:** és un paquet cohesionat que consisteix en un tipus concret de metadada. Descriu les regles que marquen el comportament de l'objecte; les referències a aquests objectes són fetes com a instàncies d'una classe. Una classe té tant una interfície com una estructura. La interfície descriu com es pot interaccionar amb la classe i la instància a través de mètodes, mentre l'estructura descriu com es divideix la dada en camps dintre de la instància. Una classe és el tipus més específic d'un objecte en relació a una capa en concret.
- **Programació orientada a objectes:** és un paradigma de programació que intenta proporcionar un model de programació basat en objectes que contenen dades i procediments associats coneguts com a mètodes. Aquests objectes, que solen ser instàncies de classes, són un tipus abstracte de dades que encapsulen (amaguen) tant les dades com les funcions per a accedir-hi.
- **Mètodes:** mena de subrutina en un llenguatge orientat a l'objecte.
- **XML:** és un format universal per a dades i documents estructurats. Els fitxers XML tenen una extensió de fitxer de xml. A l'igual que HTML, XML utilitza etiquetes (paraules delimitades pels caràcters $< i >$) per a estructurar les dades del document.
- **Middlware:** es defineix com la capa de programari que es troba entre el sistema operatiu i les aplicacions del sistema. El principal objectiu del programari intermediari és ajudar a resoldre els problemes de connectivitat i interoperabilitat entre aplicacions, servint de traductor entre diferents tecnologies i protocols. És a dir, que qualsevol aplicació (independentment del seu origen) es pugui executar sota qualsevol sistema operatiu o maquinari, facilitant així el desenvolupament de la mateixa i amagant detalls de programació de baix nivell.

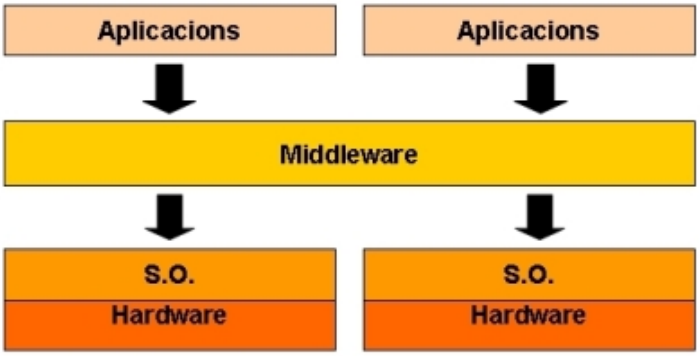

Figura 21: Esquema de la ubicació del programari intermediari (*middleware*).

Estrictament parlant el programari intermediari no és imprescindible pel correcte desenvolupament d'un procés d'integració però sí que és cert que la seva utilització simplifica molt aquests processos. Una aplicació comú per a un programari intermediari és permetre que programes desenvolupats per accedir a determinades bases de dades puguin accedir a altres bases de dades. També és comú el seu ús com a servei de missatgeria, permeten així que diferents aplicacions es puguin comunicar fàcilment.

- **API:** és una interfície que especifica com diferents components de programes informàtics haurien d'interaccionar. Dit d'una altra manera, és un conjunt d'indicacions, quant a funcions i procediments, ofert per una biblioteca informàtica o programoteca per ser utilitzat per un altre programa per interaccionar amb el programa en qüestió. O, dit encara d'una altra manera, és un conjunt de declaracions que defineix el contracte d'un component informàtic amb qui farà ús dels seus serveis.

- **HTTP:** estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia a la web. Segueix un model client-servidor on el client (generalment un navegador) inicia la petició d'informació establint una connexió TCP/IP al port 80 d'una màquina remota. El servidor espera una petició (consulta o request) amb el mètode i la versió del protocol utilitzat: p. ex. "GET / HTTP/1.2". A continuació, mitjançant una semàntica específica, s'envien una sèrie de capçaleres amb unes determinades extensions que donen metainformació sobre el tipus de document multimèdia demanat, estat de connexió, etc. a un recurs d'Internet, la referència al qual es fa mitjançant les URI (Universal Resource Identifier), com a lloc mitjançant URL (Universal Resource Locator) o com a nom mitjançant URN (Universal Resource Name).
  L'HTTP defineix vuit maneres (a vegades anomenats "verbs") d'indicar l'acció destijada que s'ha de dur a terme sobre el recurs. Les principals accions que durem a terme en aquest projecte seran:

  - **GET:** sol·licita una representació del recurs especificat. GET no s'ha d'usar per operacions que tinguin efectes col·laterals, com ara accions sobre aplicacions web. La raó d'aquesta restricció és que molts robots o web crawlers fan servir GET arbitràriament, sense tenir en compte els efectes col·laterals que les seves peticions poden causar.
  - **POST:** envia dades per a ser processades (p. ex, en un formulari HTML) a un recurs específic. Les dades s'inclouen en el cos de la petició. Això pot implicar la creació d'un nou recurs o l'actualització d'aquest si ja existeix (o ambdós).
  - **PUT:** afegeix una representació del recurs especificat.
  - **DELETE:** esborra el recurs especificat.

  La resposta del servidor és un Acknowledge seguit del fitxer demanat, un missatge d'error, etc. El mètode GET és el més comú i permet fer lectures de pàgines, però també existeixen POST, PUT, etc.

15.2 Avantatges i Inconvenients de la Metodologia *Agile*

15.2.1 Avantatges

- Permet realitzar canvis després de la fase de planificació.
- El client intervé en el procés, dóna opinions i sol·licita canvis als resultats que s'entreguen progressivament.
- Es realitzen entregues parcials en períodes curts de temps, s'avaluen i es realitza una retroalimentació amb el client.
- Les proves al final de cada període permeten corregir errors que no van ser previstos en la fase de planificació.
- Degut a que es prova contínuament, el producte pot ésser llençat al final de cada període, escurçant així la data de llançament.
- Funciona bé en projectes d'innovació.

15.2.2 Inconvenients

- Inicialment el projecte no té un pla, això pot generar que al final, el producte sigui diferent al que es va plantejar.
- Aquesta metodologia no es centra en la documentació.
- Al tenir iteracions molt llargues existeix el risc de que la solució plantejada a l'inici de l'etapa sigui diferent.
- L'avaluació dels riscs és complexa.
- Excessiva flexibilitat per alguns projectes.

15.3 Funcionalitats BackEnd i FrontEnd

La part de *backend* respresenta la capa d'accés a dades d'un *software* o qualsevol dispositiu, que no és directament accessible pels usuaris. A més, conté la lògica de l'aplicació (explicada en el següent punt) que gestiona aquestes dades. El *backend* també accedeix al servidor, que és una aplicació especialitzada que entén la manera de com el navegador sol·licita les coses.

La part de *frontend* representa la part d'un programa o dispositiu en la qual un usuari pot accedir directament. Són totes les tecnologies de disseny i desenvolupament web que corren al navegador i que s'encarreguen de la interactivitat amb els usuaris.

La web està composta per inumerables documents que estan connectats entre sí, a través d'enllaços. Quan es vol entrar a algun lloc com per exemple youtube escrivim la URL www.youtube.com a la barra del navegador, això vol dir que estem sol·licitant que ens mostrin una pàgina web.

En el següent pas, la nostra computadora verifica quin servidor de *software* està corrent el lloc. Aquí el servidor mira quin tipus de petició és la que estem realitzant. Si entrem a `https://www.youtube.com/watch?v=BlWNRk7m0pk` representa una petició GET que obté el lloc. En aquest cas, ha de connectar-se a una base de dades per obtenir el vídeo.

Algunes vegades no es requereix connexió a la base de dades, per exemple: a l'entrar al lloc `https://platzi.com/login` no fa falta, ja que quan s'inicia sessió es fa una petició de tipus POST que es connecta a la base de dades per a validar els accessos i el tipus de conta o subscripció que tenim, i amb això el backend retorna la resposta al servidor per a que la impulsi al navegador.

Finalment, entra el *frontend* que rep la informació que li ha passat el *backend* i la posa a la interfície del lloc que li pertoca, que en aquest cas cas seria el perfil de l'usuari. Aquest fet és el que finalment es mostra al navegador.

15.4 Programació per capes.

La programació per capes és un model de desenvolupament *software* on l'objectiu primordial és la separació (desacoblament) de les parts que composen un sistema *software* o també una arquitectura client-servidor: capa de presentació, capa de negoci i capa de dades. D'aquesta forma, per exemple, és senzill i mantenible crear diferents interfícies sobre un mateix sistema sense requerir cap canvi en la capa de dades o lògica.

L'avantatge principal d'aquest estil és que el desenvolupament es pot dur a terme en diversos nivells i, en cas de que sobrevingui algun canvi, sols afectarà al nivell requerit sense tenir que revisar entre el codi font d'altres mòduls, donat el cas en que s'haurà reduït l'acoblament informàtic fins una interfície de pas de missatges.

A més, permet distribuir el treball de creació d'una aplicació per nivells; d'aquesta manera, cada grup de treball està totalment abstret de la resta de nivells, de forma que n'hi ha prou amb conèixer l'API que existeix entre nivells.

En el disseny de sistemes informàtics actual es solen usar les arquitectures multinivell o programació per capes. En aquestes arquitectures a cada nivell se li confia una missió simple, el que li permet el disseny d'arquitectures escalables (que poden ampliar-se amb facilitat en cas de que les necessitats augmentin).

15.4.1 Capes i nivells.

- **Capa de presentació:** la que veu l'usuari (també es denomina « capa d'usuari »), presenta el sistema a l'usuari, li comunica la informació i captura la informació de l'usuari en un mínim de procés (realitza un filtrat previ per a comprovar que no hi ha errors de format). També es conegut com a interfície gràfica i ha de tenir la característica d'ésser « amigable » (comprensible y fàcil d'usar) per al usuari. Aquesta capa es comunica únicament amb la capa de negoci.

- **Capa de negoci:** és on resideix els programes que s'executen, es reben les peticions de l'usuari i s'envien les respostes després del procés. Es denomina capa de negoci (i inclús de lògica del negoci) perquè és aquí on s'estableixen totes les regles que s'han de complir. Aquesta capa es comunica amb la capa de presentació, per a rebre les sol·licituds i presentar els resultats, i amb la capa de dades, per sol·licitar al gestor de base de dades emmagatzemar o recuperar dades d'ell. També es consideren aquí els programes d'aplicació.

- **Capa de dades:** és on resideixen les dades i és la encarregada d'accedir als mateixos. Està formada per un o més gestors de bases de dades que realitzen tot l'emmagatzemament de dades, reben sol·licituds d'emmagatzemament o recuperació d'informació des de la capa de negoci.

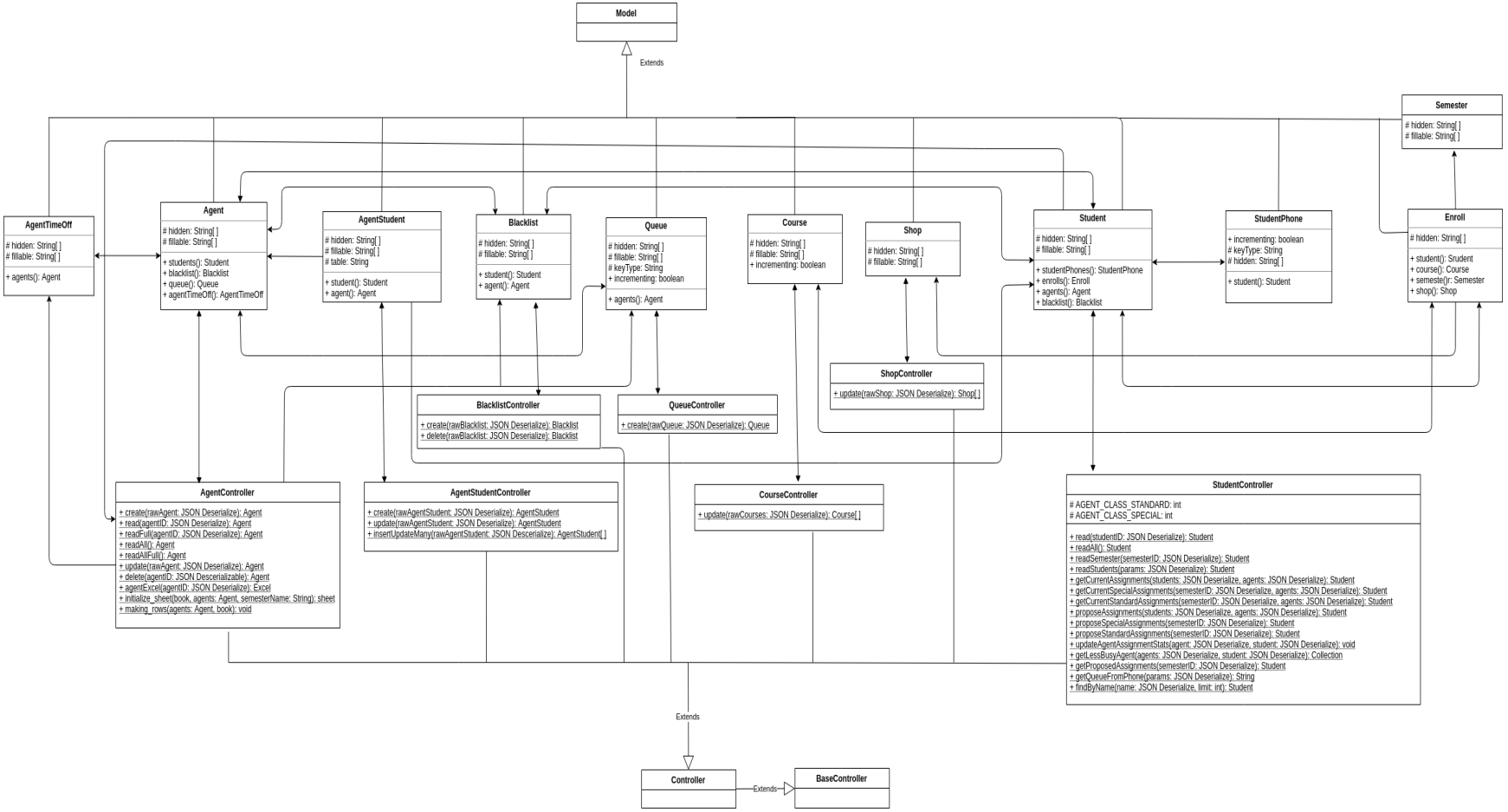

Figura 22: Diagrama UML sobre els Models i els Controladors de l'aplicació.

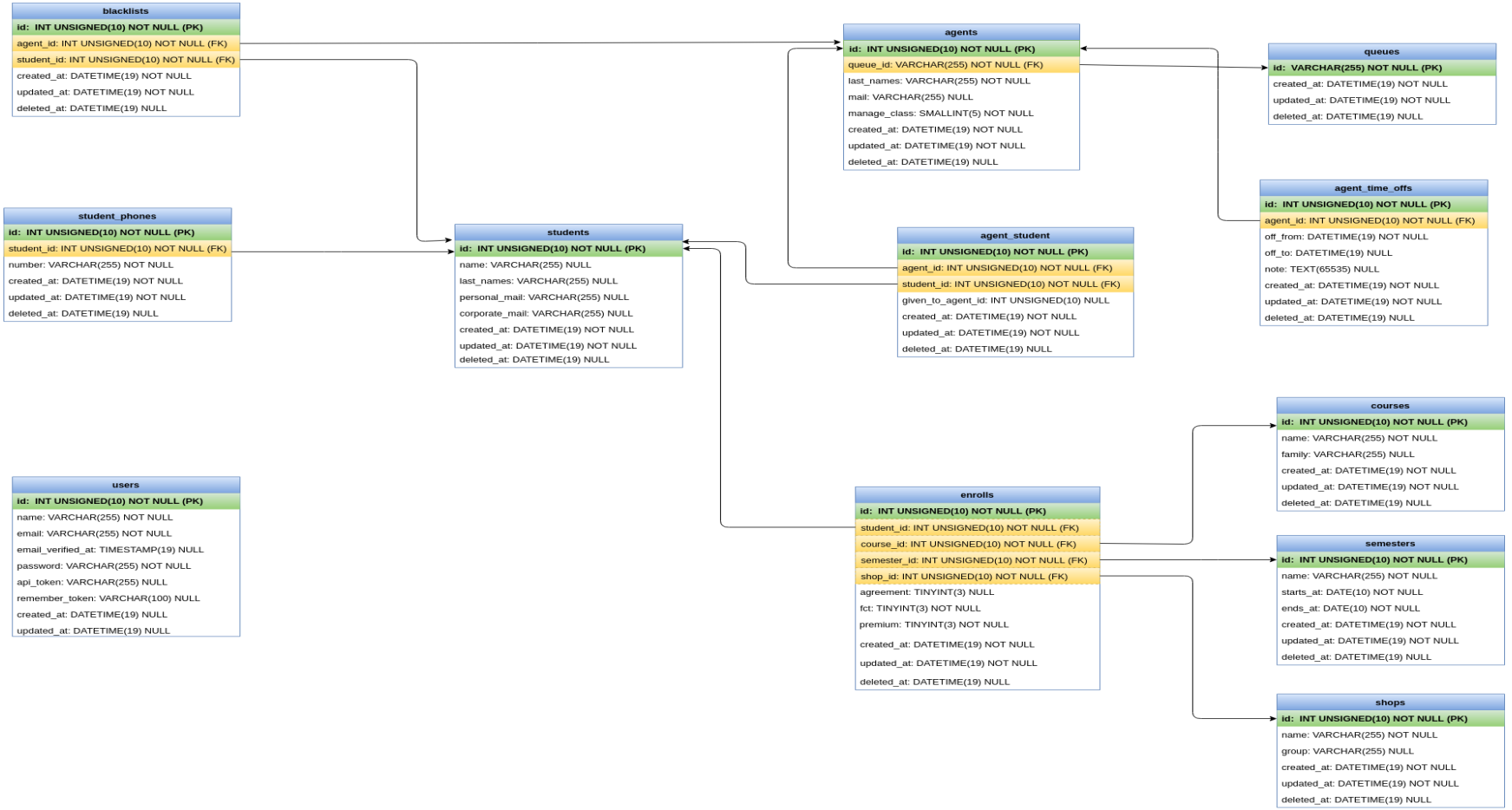

Figura 23: Diagrama UML sobre la base de dades de l'aplicació.